# LogiCORE IP Digital Pre-Distortion v5.0

DS856 June 22, 2011 **Product Specification**

## Introduction

Pre-distortion negates the non-linear effects of a power amplifier (PA) generated when transmitting a wide-band signal. Pre-distortion allows a PA to achieve greater efficiency by operating at higher output power while still maintaining spectral compliance, reducing system capital and operational expenditure.

The solution is targeted for base stations used in third and fourth generation (3G/4G) mobile technologies and beyond. It is a combination of hardware and embedded software processes that between them realize pre-distortion correction along with features that make for a fully engineered, practical, robust and self-contained solution. It is configurable both in feature selection and in usage to support a variety of clocking and resource requirements.

## Features

- Algorithms
  - DPD correction with up to 33 dB of ACLR improvement
  - Pre-distortion correction architecture selection for cost-performance trade-off
  - Dynamics options
  - TDD support with automatic data selection
  - Quadrature modulator correction
  - PA saturation (overdrive) detection
  - Signal capture and analysis
  - Easy integration and evaluation using the Debug Interface utility [Ref 2]
- Physical Configuration Parameters
  - Selection of correction architectures of increasing performance/complexity
  - Selection of polynomial order of 5 or 7
  - Selection of one, two, four or eight transmit antennas
  - Clock to sample rate ratios from one to four
  - Optional quadrature modulation correction for either the transmitter or feedback path receiver
  - Optional hardware acceleration of coefficient estimation and signal alignment
- Interface Options
  - Real IF feedback signal sampled at twice the pre-distortion sample rate with arbitrary IF frequency (optimal performance option)
  - Real IF feedback signal sampled at one times the pre-distortion sample rate with arbitrary IF frequency
  - Zero-IF complex baseband feedback signal sampled at one times the pre-distortion sample rate with integrated QMC.
  - Optional feedback path support for buffered ADC support.

| LogiCORE IP Facts Table | |
|---|---|
| **Core Specifics** | |
| Supported Device Family(1) | Virtex-7, Kintex™-7, Artix™-7, Zynq™-7000, Virtex-6 |
| Supported User Interfaces | AXI4-Lite & AXI4-Stream |
| Configuration | See Resource Utilization and Performance |
| **Provided with Core** | |
| Documentation | Product Specification |
| Design Files | Netlist |
| Example Design | Not Provided |
| Test Bench | Not Provided |
| Constraints File | See Using Constraints |
| Simulation Model | See Running Simulation |
| **Tested Design Tools** | |
| Design Entry Tools | ISE v13.2 |
| Simulation(2) | Mentor Graphics ModelSim |
| Synthesis Tools | Not Provided |
| **Support** | |
| Provided by Xilinx, Inc. | |

1. For a complete listing of supported devices, see the release notes for this core.
2. For the supported version of the tools, see the ISE Design Suite 13: Release Notes Guide

# Applications

An easy-to-use software interface allows configuration, single-stepping and continuous automatic operation while providing access to signal measurements, data, diagnostic and status information.

## Usage Overview

This section briefly summarizes a sequence of events for successful incorporation of DPD into a radio unit FPGA. Later sections provide the necessary detail.

### Instantiation

1. The DPD component is added into the user's HDL code with appropriate clocks and interfacing and the optional Debug Interface.
2. DPD is placed after Crest Factor Reduction (CFR) in the transmit chain.
3. The design is compiled.
4. A SW environment for reading and writing the host interface is established. A ready made environment is available using the MATLAB® based Debug Interface for quick evaluation on the target platform.

### Basic Operational Checks

1. Read the addresses specified in Table 14 from the host interface; the stated default values should be seen.
2. Execute (for example) the RESET_COEFFICIENTS control mode (see Host Interface and SW Control Modes) to check termination with successful status.

### Software Setup and Signal Validation

1. Set up DPD parameters as described in Setting DPD Parameters.
2. Read the DPD monitors detailed in Table 15.
3. Determine whether the values for the transmit and receive powers are as expected.
4. Perform required operations as detailed in Signal Analysis to ensure that the signal inputs conform to the recommendations in Factors Influencing Expected Correction Performance.

### Pre-distortion Operations and Achieving Performance

1. Adjust DPD parameters and external setup with the aid of the single-stepping commands (see Single Stepping), external measurements, signal analysis operations and interpretation of diagnostics as required.
2. In certain circumstances it is desirable to initialize the DPD and QMC coefficients using a known good set of coefficients. The Xilinx DPD solution allows for the reading and loading of coefficients to facilitate the initialization feature (see Reading and Loading Coefficient Set).
3. Run the DCL (see Running the DCL) with diagnostic monitoring to experience the full operational capability of DPD.

# Functional Description

## Mathematical Foundation

Digital Pre-Distortion (DPD) acts on transmitted data to cancel the distortion in the PA by implementing an inverse model of the amplifier. In the conceptual view of Figure 1, the pre-distortion function is applied to the sequence of (digital) transmitted data x(n). It models the non-linearity of the PA.

The processes involved are the formulation of the model on which the pre-distortion function is based. Estimation of its parameters is based on samples of the PA input and output. To separate the linear effect of the PA and the circuitry that drives it, estimation is based on the *aligned* PA output y(n). The alignment process matches the amplitude, delay and phase variations of $y_0(n)$ to z(n). The predistorter is then dedicated to only modeling the non-linear effects for which it is intended. *Alignment* and *estimation* blocks are depicted in Figure 1.

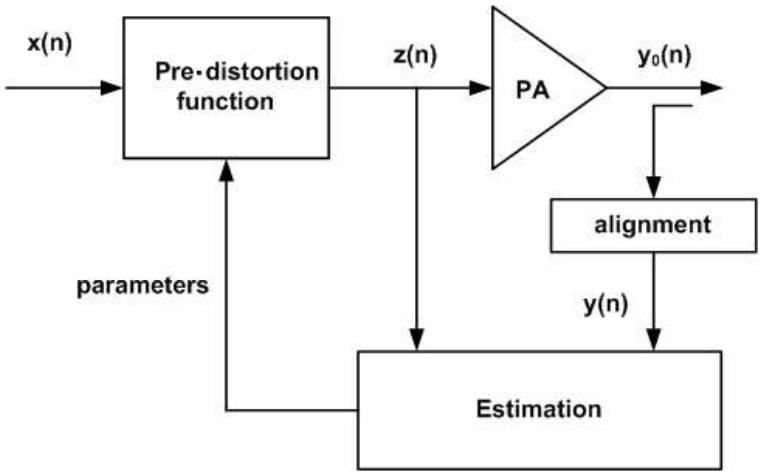


*Figure 1:* **DPD Algorithmic View**

### Volterra Series

The Volterra Series is a well known expansion for non-linear functions in space and time. Here the "space" dimension is the (complex) signal value and the discrete form in time is used. It is a suitable starting point for a pre-distortion function. The Volterra Series is given in Equation 1; {h} are constants.

$$z(n) = \sum_{i} h_1(i)x(n-i) + \sum_{i1}\sum_{i2} h_2(i1,i2)x(n-i1)x(n-i2) + \sum_{i1}\sum_{i2}\sum_{i3} h_3(i1,i2,i3)x(n-i1)x(n-i2)x(n-i3) + \ldots..$$

*Equation 1*
*Volterra Series*

### Physical Volterra Series

Without loss of generality, the Volterra series can be written as

$$z(n) = \sum_{q=0}^{Q-1} F_q x(n-q)$$

*Equation 2*
*Nonlinear Moving Average Form of the Volterra Series*

In Equation 2, $F_q$ can be called *memory terms*. If Equation 2 is to model a power amplifier, it must conform to the boundary condition that when the signal amplitude $|x|$ is small, the model reduces to a linear time invariant system (because the PA is linear for small signals). A sufficient condition for this is that the memory terms depends only on samples of signal *magnitude*, $|x(n - p)|$ with $p <= q$.

### Volterra Product Selection

There are many possible terms in the Volterra series, and a practical solution involves judicious selection among them.

Within the physical Volterra series, $F_q$ is in general a function of all samples of $|x(n - p)|$ with $p <= q$.

Inspection of the physical Volterra series reveals that

$$F_q = \sum_{\{i\}} a_{\{i\}q} f_{\{i\}q}$$

*Equation 3*
*Series Expansion of the Memory Terms*

where the set {i} covers the number of indices required to form the terms. Comparison with Equation 1 shows that the basis functions $f_{\{i\}q}$ are of the form $|x(n - s)|^a |x(n - t)|^b |x(n - v)|^c \ldots$ . These can be called *Volterra products*.

The simplest possible form for $f_{\{i\}q}$ is unity (a = b = c …. = 0). In this case the series is an FIR filter.

Xilinx DPD versions 1 and 2 used the well known Memory-Polynomial (MP) model. In this model, (i) is one index k running from 0 to K - 1 and $f_{kq} = |x(n - q)|^k$. This corresponds to selecting diagonal Volterra series terms. The *memoryless* pre-distortion model is a special case of this with Q = 1.

In this DPD version, there are options to use the Memory-Polynomial model, but also to configure for models based on a more general selection of terms. With these configurations, improved correction performance, particularly for wideband signals, is observed. The *pre-distortion correction architecture* is an option for core generation and software configuration. Four possible architectures - called A, B, C and D - can be selected, having increasing complexity of diagonal and off-diagonal memory terms. In addition the polynomial order can be selected to be 5 or 7. The polynomial order is the maximum value of a,b,c ... appearing in the Volterra products used.

### Estimation of the Coefficients

The objective of *pre-distortion estimation* is to choose coefficients $a_{\{i\}q}$ such that the PA output $y_0(n)$ is as close as possible to x(n).

This is done by capturing a sequence of L samples of the PA input and output and using Equation 2 in a reverse sense to make a linear equation for the $a_{\{i\}q}$ for each sample. A least-squares solution can then be found for the $a_{\{i\}q}$.

The least-squares problem for $a_{\{i\}q}$ can be expressed in matrix form as

$$Z = UA$$

*Equation 4*
*Parameter Estimation in Matrix Form*

where Z is a column vector of the signal samples z(n), A is a row vector of all the $a_{\{i\}q}$ and the rows of U are the elaboration of all the $(|x(n - s)|^a |x(n - t)|^b |x(n - v)|^c \ldots)x(n - q)$ in the model for each x(n) = y(n), the samples of the aligned PA output.

Equation 4 can be solved by pre-multiplying each side by $\mathbf{U^H}$, the Hermitian transpose of U, to give

$$VA = W$$

*Equation 5*
*System to be Solved for the Pre-distortion Coefficients*

In Equation 5, $\mathbf{V = U^H U}$ and $\mathbf{W = U^H Z}$. It is a linear system whose solution is the best least-squares estimate for $a_{\{i\}q}$ over the sample length L.

An entirely new set of coefficients can be obtained from each new data capture. Alternatively, the coefficient can be iterated with the Damped-Newton method.

The solution to **VA = W** can be expressed as **A = V \ W**. Within this notation, the Damped-Newton method iterates A according to $\mathbf{A_{n+1} = A_n + \mu\, V \setminus WE}$, where $\mathbf{WE = U^H E}$ and $\mathbf{E = Z - UA_n}$. It iteratively acts to minimize an error vector E, which is the difference between the transmitted samples and the predicted transmission based on the inverse model over the receive samples. The damping factor μ is an adjustable parameter.

This is a general mathematical method which, when applied to DPD, improves immunity to noise and distributes noise that is non-uniform in time over many updates, thus making the typical instantaneous behavior equal to the mean behavior.

The Damped-Newton iteration is used only when the power is stable, as it cannot react to fast dynamics.

## System Features

### Hardware Description

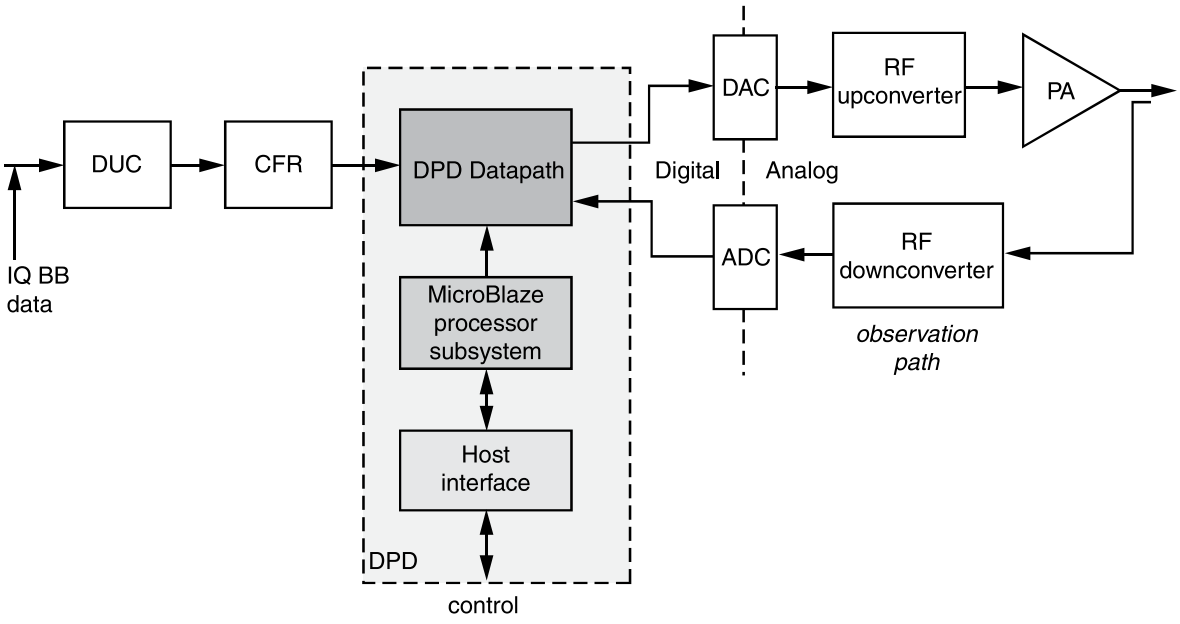


*Figure 2:* **Xilinx DPD HW Block View**

Figure 2 shows that DPD is placed after CFR in the transmit signal chain. DPD operates at the DPD sample rate $f_s$. The selection of $f_s$ is discussed in Factors Influencing Expected Correction Performance. The signal is converted into the analog domain by the DAC component. There can be further interpolation stages between DPD and the DAC. Moreover, there can be digital mixing for a single DAC superheterodyne transmitted or IQ DACs for a direct conversion transmitter. Whatever choices are made, based on system-level considerations, the net result is that the IQ

data from DPD eventually appears as modulation of an RF carrier wave at the PA. To estimate pre-distortion coefficients, a sample of the PA output is fed back via the observation path and must finally be presented to the estimator as IQ samples at $f_s$. For optimal sampling bandwidth, either direct RF downconversion followed by an IQ ADC pair or a heterodyne mixing to $f_s/4$ and an ADC sampling at $2f_s$ should be used. Sub-optimal sampling bandwidth can also lead to good pre-distortion performance, depending on the signal. DPD supports a single ADC at $f_s$ and variable feedback IF frequency. For single ADC architectures, digital downconversion is required, and this is performed on the data prior to the estimation processing.

Within DPD, the datapath contains resources to deal with the real-time processing required for up to eight individual antennas. A single MicroBlaze™ processor sub-system performs estimation and supporting algorithms – it is shared between the antennas. Details of the system are given in the following sections.

The host interface is a shared-memory data and message-passing subsystem.

### HW-SW Co-design

Xilinx DPD is a combination of HW and SW processes that between them realize the PA distortion inverse model and the estimation algorithm, as described in the Mathematical Foundation section, along with features that make for a fully engineered, practical, robust and self-contained solution.

Figure 3 depicts the main elements of the DPD solution. The HW processes are contained within the datapath and the SW processes are run in the MicroBlaze processor code. There are also Quadrature Modulator Correction (QMC) and Overdrive Detection (ODD) SW processes not indicated in the diagram.

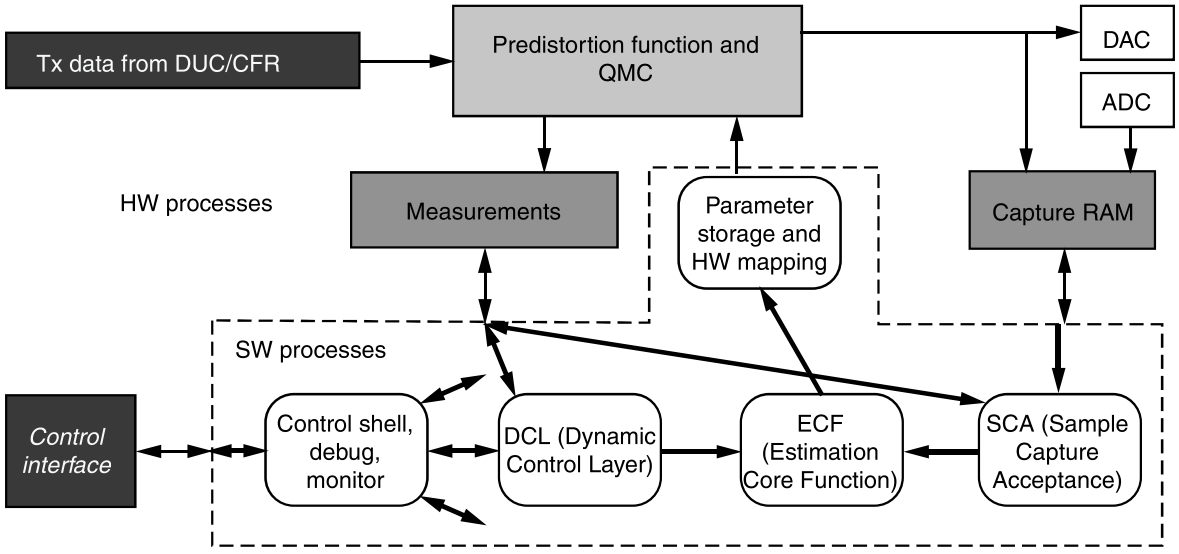


*Figure 3:* **HW Datapath and Major SW Processes**

### Capture RAM and Estimation Core Function (ECF)

The capture RAM collects L complex samples of the transmitted data and the appropriate samples from the observation path ADC, depending on the receiver configuration.

The ECF performs digital downconversion and then alignment and coefficient estimation as outlined in Mathematical Foundation. The coefficients are mapped into an efficient structure that avoids direct computation of the selected Volterra products in hardware.

The ECF performs checks on the signals and reports any adverse status (see Host Interface and SW Control Modes) if there is an issue, in which case, the pre-distortion function parameters are not updated.

### Measurements Block and Sample Capture Acceptance (SCA)

The capture RAM is able to capture tens of microseconds of data. To obtain optimum estimation results, the capture needs to contain data that is representative of the signal over the lifetime of the coefficients, a time much longer than the capture length. In particular, if the capture is taken during a period of time in which the signal amplitude is small, for example during the low period of a data pulse, the estimation will be poor for higher amplitude signals that might occur over a longer timescale. This is similar in principle to fitting a polynomial curve to points in a scatter diagram. The function beyond the region where there are data points is an extrapolation, yielding unpredictable behavior.

In some cases there is a frame structure with a known good period that can be used for estimation. For this case, DPD can be configured to allow the capture to be triggered by an external sync pulse, with a programmable delay. Otherwise, the SCA feature can be enabled.

SCA takes captures at random and applies acceptance criteria to the samples in the capture RAM to ensure that the samples are representative of the full signal. The acceptance criteria are based on the average power and statistical measures (histograms) of the transmitted data in the capture buffers, and their comparison to the average power and statistics of the transmitted signal over a defined measurement interval (typically greater than 10 ms). The *Measurements* block depicted in Figure 2 provides the real-time processing of the signals required to form the powers and histogram. This data is also available to the user via the control interface to aid system debug and monitoring.

When SCA is running, TDD signals are automatically catered for – a capture taken in the uplink period fails and only data transmitted in the downlink period is ever used.

### Dynamic Control Layer (DCL)

DCL is the mechanism by which DPD adapts to power dynamics encountered in a cell due to call load, reconfiguration or other factors.

DPD must take into account how the correction performance of a real PA depends on the output power dynamic after some parameters are estimated. If the power steps up after estimation, the correction typically gets worse. It might also get worse when the power steps down. Therefore it is not sufficient to update the coefficient repeatedly, even if done rapidly, because every time there is a step-up, the correction is poor until re-estimation. Repeated step-up events inevitably cause a high integrated out-of-band emission, whatever the estimation rate.

The Xilinx DPD DCL operates by retaining a memory of the behavior of the PA in various signal conditions held in one or more stored parameter sets. The parameters applied at any one time are determined by criteria that depend on the momentary signal condition in relation to the signal conditions associated with the stored parameter sets. The decision process for changing the applied parameters uses the power measurement block in the datapath and happens at the rate set by the METER LENGTH parameter. A typical rate is 10 ms. Single-set and multiple-set modes are available (see DCL Parameters, page 30). The latter should be selected for PAs where the pre-distortion correction gets worse when the power steps down. Specifically, the user should test by running the DCL in single set mode whilst transmitting initially at maximum power. If the spectral performance becomes unacceptable when the power is backed-off, multiple set mode can be used.

### Multipath Handling

The estimation process can update only one path at a time. The multipath DCL attends to each path in turn and estimation is performed only if the power for the port being examined satisfies the criteria for coefficient update (the same criteria as with the single-port design).

In the limiting case where only one path ever satisfies the criteria for coefficient update, that path is continuously re-estimated. This also means that if one path fails, DPD operates correctly on the others.

## Quadrature Modulator Correction (QMC)

Extending the Xilinx DPD solution to support Quadrature Modulator Correction (QMC) involves the addition of a simple MUX, adding software functions to the MicroBlaze processor C-code to compute the QMC update, and the actual QMC logic (for TX QMC only). This solution requires no special waveforms, training sequences or additional RF hardware; it is fully integrated into the DPD solution and is able to operate virtually transparently while DPD is operational.

The use of QMC is only supported at either the TX or RX side. It cannot be applied to both TX and RX side simultaneously because the QMC, as implemented, cannot distinguish between QM imperfections on the TX side versus imperfections on the RX side if direct conversion were used on both. The location of the QMC does not affect the DPD update which always uses TX samples captured from after the DPD filter. The RX samples are aligned to the TX samples (using the TX samples as a reference). The aligned samples are then processed by the Least Squares estimator to compute DPD coefficients.

### *TX QMC support*

Figure 4 shows at the top level the added hardware required to support TX QMC in the DPD design (the additions are shown in dotted RED lines).

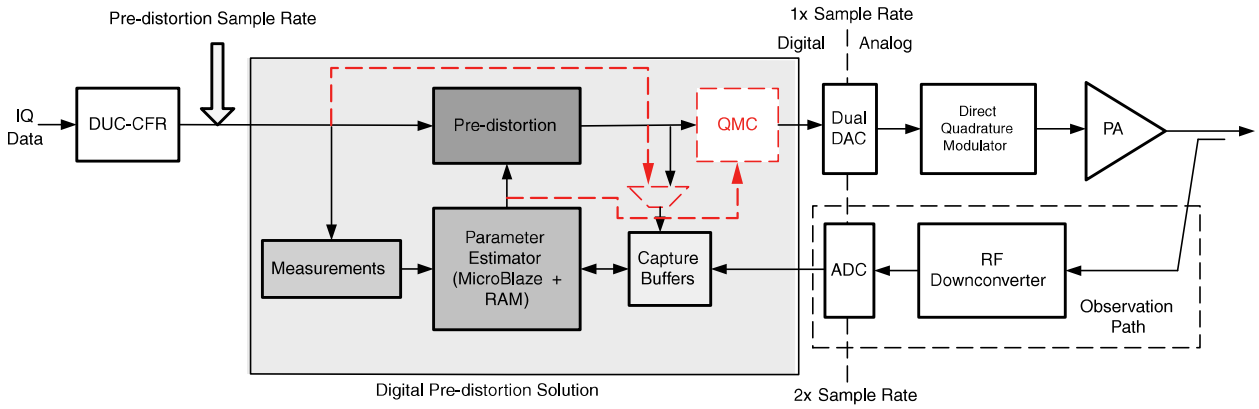


*Figure 4:* **DPD with Integrated TX QMC Architecture**

Figure 5 outlines the steps required to produce the coefficients used in DPD and QMC.

A QMC update uses TX samples captured from before the DPD filter. The RX samples are aligned to the TX samples (using the TX samples as a reference). The aligned samples are then processed by the QMC update processor to update the QMC coefficients.

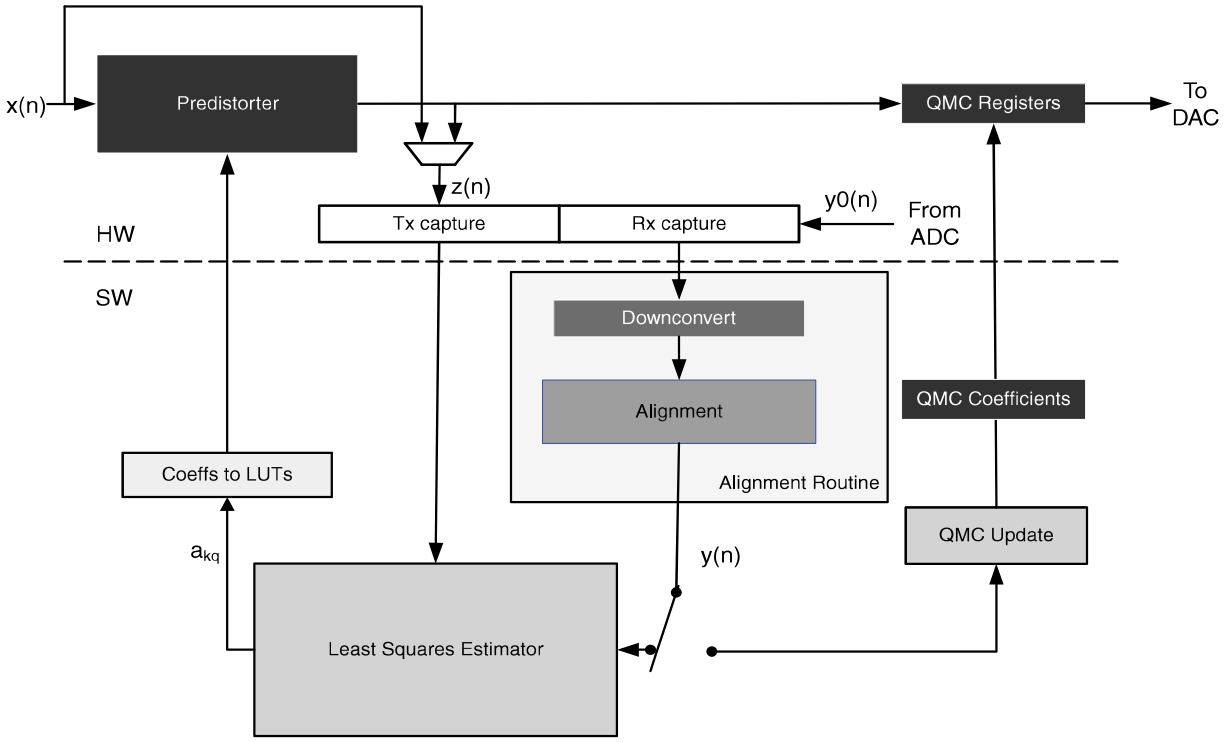


*Figure 5:* **DPD Estimation Core Function (ECF) Processes with integrated TX QMC**

### *RX QMC support*

Figure 6 shows at the top level the added hardware required to support RX QMC in the DPD design (the additions are shown in dotted RED lines).

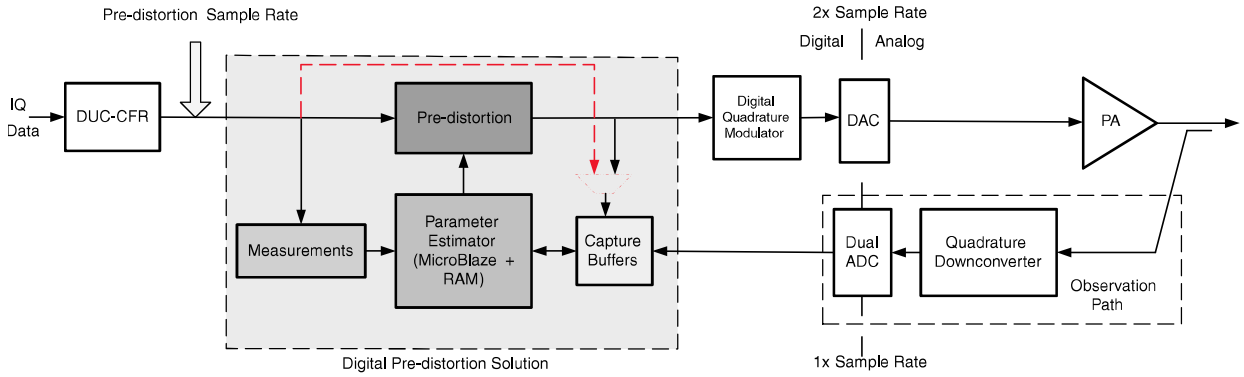


*Figure 6:* **DPD with Integrated RX QMC Architecture**

Figure 7 outlines the steps required to produce the coefficients used in DPD and QMC.

A QMC update uses TX samples captured from before the DPD filter. The RX samples are aligned to the TX samples (using the RX samples as a reference). QM corrections are applied to the RX samples as part of the alignment routine. The aligned samples are then processed by the QMC Update processor to update the QMC coefficients.

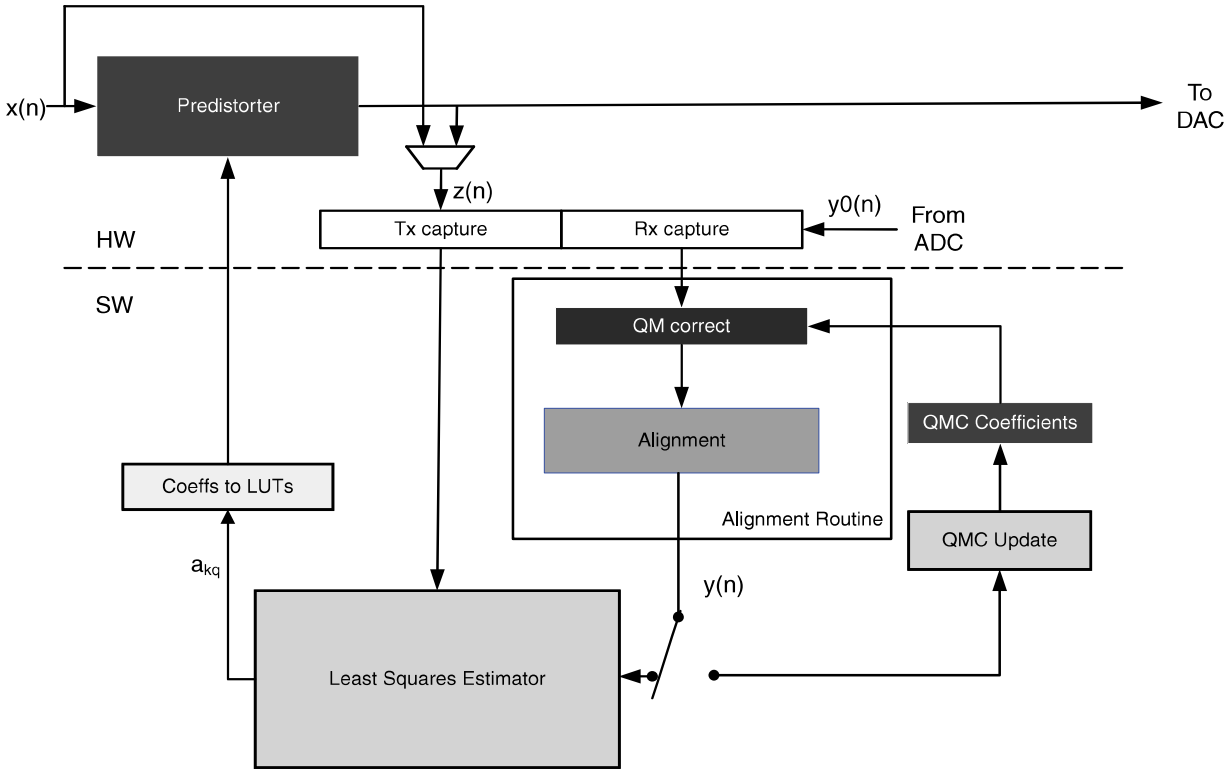


*Figure 7:* **DPD Estimation Core Function (ECF) Processes with integrated RX QMC**

When the QMC SW function is enabled, its parameters are estimated iteratively from sample captures that are interleaved with pre-distortion correction estimation.

### Overdrive Detection (ODD)

Setting the maximum drive level for an RF power amplifier under operational conditions can be a non-trivial task. The use of digital pre-distortion allows the PA to be driven such that the peaks of the predistorted waveform can be driven very close to the saturation point of the PA. However any form of pre-distortion begins to fail when the peaks of the waveforms enter the saturating region of the amplifier.

DPD contains a function that dynamically detects when the signal is being driven too hard. The method is predictive, which means that an overdrive condition can be detected before it actually occurs. The method produces a "soft" overdrive metric so system performance can be tuned to meet the needs of a specific installation.

After each pre-distortion coefficient estimation, ODD computes an overdrive metric. The overdrive metric is compared to an overdrive threshold (which is a user parameter), and the result is used to report an overdrive status for that estimation, which can be monitored. There is also an option to prevent the coefficients from being used whenever overdrive is detected.

Figure 8 shows an example of the overdrive metric while sweeping the RF drive level along with the corresponding reduction in ACLR achieved by DPD while transmitting a WCDMA waveform. In the region where the normalized RF drive level is below -20 dB, there is very little distortion from the amplifier, hence no improvement from the use of DPD. As the RF drive level is increased, the amount of distortion created by the PA increases. DPD is able to reduce the distortion (by around 22 dB in this case) until the PA begins to be overdriven. Distortion caused by overdrive cannot be compensated by any form of pre-distortion due to the saturating nature of the amplifier. The

overdrive metric indicates that overdrive is beginning to occur when the normalized RF drive level reaches around 0 dB. Increasing the RF drive level above this point by just a small amount has a drastic effect on the ability of the DPD to correct the induced distortion (this is shown in red in Figure 8).

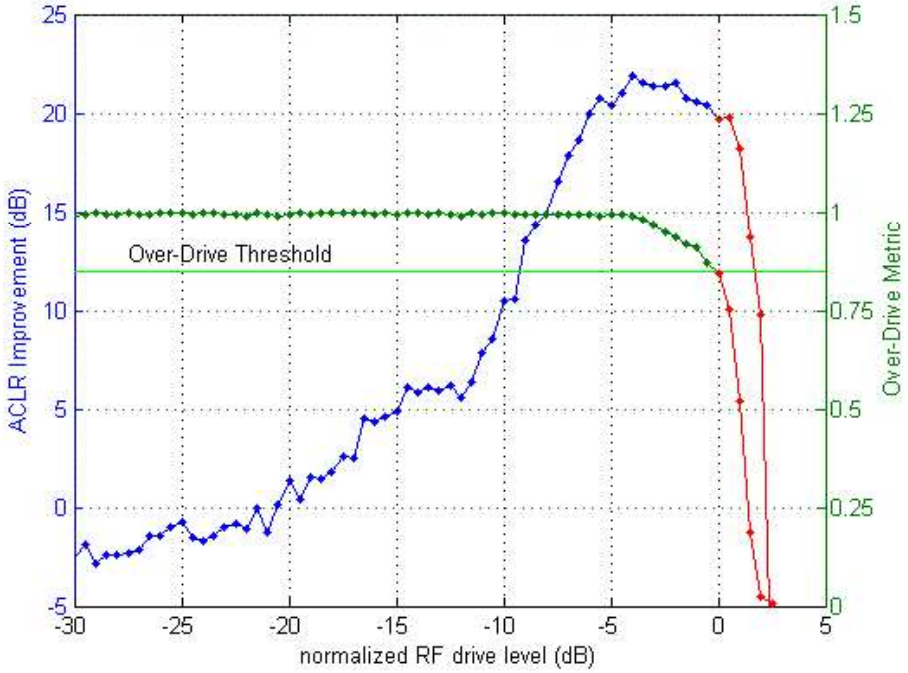


*Figure 8:* **ODD Example Characterization**

# Core Instantiation and Configuration

## DPD GUI Parameters

The core configuration window is shown in Figure 9. When the mouse hovers over each parameter, tool tips appear with a brief description, as well as feedback about how their values or ranges are affected by other parameter selections.

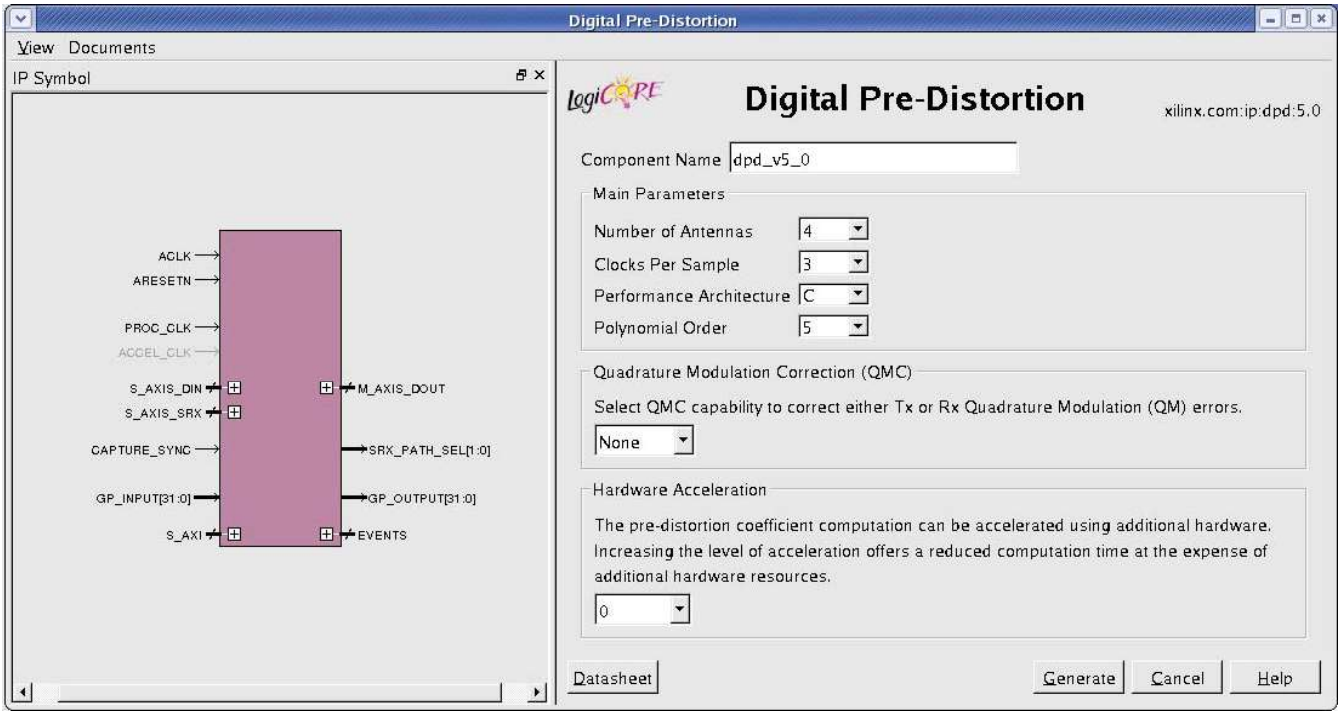


*Figure 9:* **DPD Core GUI**

### Component Name

Enter the name of the core component to be instantiated. The name must begin with a letter and be composed of the following characters: a to z, A to Z, 0 to 9, and '_'.

### Number of Antennas (TX)

One, two, four or eight transmitter antenna paths can be included in an instance of this IP core. Each core instance includes only one MicroBlaze processor. Multiple core instances can be used for improved dynamics performance; for example, two core instances of two antenna support each instead of one core instance of four antenna support.

### Clocks Per Sample (CLOCKS_PER_SAMPLE)

This core offers the user the option to select different hardware folding options by setting this parameter to either 1, 2, 3 or 4. When the parameter is set to 1, the entire design runs at the same rate as the input sample rate. This permits very high sample rates to be achieved - up to 300 Msps on Virtex®-6, capable of handling 60 MHz of signal bandwidth. When the parameter is set to 2, 3 or 4, various portions of the design take advantage of hardware folding to offer a resource saving in the hardware implementation. The Resource Utilization section provides a subset of the resource utilization data for different parameter combinations of the core. Similarly, the IP Timing Performance provides some timing performance data. A spreadsheet with detailed resource and timing characterization data can be obtained by contacting technical support.

### Performance Architecture (ARCH)

The architecture configuration of the IP is chosen to suit performance requirements and selected device capacity. Architecture A is the memory-polynomial model (see Volterra Product Selection). Architectures B, C and D typically offer distortion correction performance improvement (but at increasing resource cost), dependent upon the radio platform and signal characteristics. Users might want to set this to be D during a performance evaluation phase if they can fit the design in their chosen devices. Users then have an option to evaluate the performance using software controls as described in Operations Guide for smaller architectures A, B and C. For example, if the performance of architecture C is satisfactory, resources can be saved by regenerating the core with option C. When architecture D and 7th polynomial order is selected there is a new software selectable architecture (D7f) that is beneficial for wide band applications. D7f is enabled by setting ARCH_SEL = 84 in the host interface.

### Polynomial Order (POLY_ORDER)

This parameter sets the maximum polynomial order, as defined in the Functional Description. When set to 7 in the GUI, either 7 or 5 can be selected through the software interface, as described in the Operations Guide.

The primary effect of setting POLY_ORDER to 7 is an increase in the number of block RAM memories with improved performance for wider bandwidth signals.

### Quadrature Modulation Correction (QMC)

This parameter, when set to Tx_QMC, provides QMC capabilities on the transmit path and requires extra hardware resources. When this parameter is set to Rx_QMC, QMC is applied on the received captured data using a software routine. This capability is available with FIFO ADC mode of operation as well.

### Hardware Acceleration (HWA)

When this checkbox is set to 1, 2 or 3, it enables increasing hardware acceleration logic within the core to accelerate the coefficient computation. This adds additional resources and should be disabled if the increased speed is not required. When this parameter is set to a non-zero value, a separate input clock `accel_clk` is available on the netlist. The IP Timing Performance provides timing data for this clock. The SW Features Timing Performance section provides details about the pre-distortion speed-up offered for various parameter combinations with different levels of acceleration. When this parameter is set to 1, it offers the same acceleration as the DPD v4.0 core `has_hwa=true` option.

## Input/Output Ports

Figure 10 displays the signal names; Table 1 defines these signals.

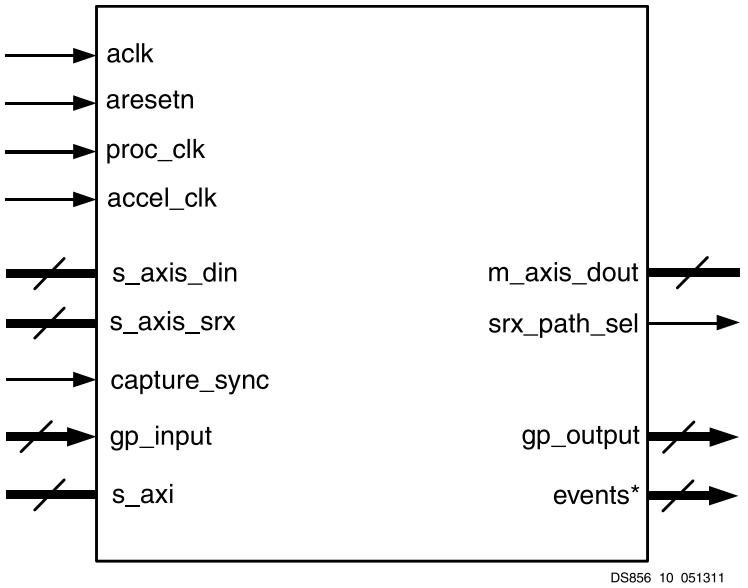


*Figure 10:* **DPD IP Symbol (Input/Output Ports)**

*Table 1:* **Top-Level I/Os for the DPD IP Core**

| Names | Direction | Description |
|---|---|---|
| aclk | Input | Rising-edge clock for datapath |
| aresetn | Input | Common active low synchronous reset for datapath, processor logic and any hardware accelerators present |
| proc_clk | Input | Rising-edge clock for MicroBlaze processor |
| accel_clk | Input | Optional rising-edge clock for hardware acceleration engines |
| **DIN Interface (Slave AXI4-Stream)** | | |
| s_axis_din_tdata[16*2*TX-1:0] | Input | 16 bit Q and I component per transmit antenna cascaded |
| s_axis_din_tvalid | Input | Valid handshake |
| s_axis_din_tready | Output | Ready handshake |
| **SRx Interface (Slave AXI4-Stream)** | | |
| s_axis_srx_tdata[16*2-1:0] | Input | 16 bit srx_din1 and srx_din0 cascaded |
| s_axis_srx_tvalid | Input | Valid handshake |
| s_axis_srx_tready | Output | Ready handshake |
| **DOUT Interface (Master AXI4-Stream)** | | |
| m_axis_dout_tdata[16*2*TX-1:0] | Output | 16 bit Q and I component per transmit antenna cascaded |
| m_axis_dout_tvalid | Output | Valid handshake |
| m_axis_dout_tready | Input | Ready handshake |
| **Control Signals** | | |
| capture_sync | Input | Signal to optionally synchronize pre-distortion capture |
| srx_path_sel[K-1:0] | Output | Control for receive path selection in multiple-path designs; K= 1, 1, 2 or 3 when TX is set to 1, 2, 4 or 8 respectively. |

*Table 1:* **Top-Level I/Os for the DPD IP Core *(Cont'd)***

| Names | Direction | Description |
|---|---|---|
| **General Purpose I/O** | | |
| gp_input[31:0] | Input | General purpose 32 bit input (used for FIFO ADC interface) |
| gp_output[31:0] | Output | General purpose 32 bit output (used for FIFO ADC interface) |
| **Host Interface (AXI4-Lite)** | | |
| s_axi_aclk | Input | Rising edge clock |
| s_axi_aresetn | Input | Active low synchronous reset |
| s_axi_wdata[31:0] | Input | Write data |
| s_axi_wvalid | Input | Write data valid handshake |
| s_axi_wready | Output | Write data ready handshake |
| s_axi_awaddr[10:0] | Input | Write address (byte addressing) |
| s_axi_awvalid | Input | Write address valid handshake |
| s_axi_awready | Output | Write address ready handshake |
| s_axi_araddr[10:0] | Input | Read address (byte addressing) |
| s_axi_arvalid | Input | Read address valid handshake |
| s_axi_arready | Output | Read address ready handshake |
| s_axi_bresp[1:0] | Output | Write response |
| s_axi_bvalid | Output | Write response valid handshake |
| s_axi_bready | Input | Write response ready handshake |
| s_axi_rdata[31:0] | Output | Read data |
| s_axi_rvalid | Output | Read data valid handshake |
| s_axi_rready | Input | Read data ready handshake |
| s_axi_rresp[1:0] | Output | Read data response |
| **events*** | | |
| event_s_din_underrun | Output | underrun status flag for s_axis_din AXI4 bus |
| event_m_dout_overrun | Output | overrun status flag for m_axis_dout AXI4 bus |
| event_s_srx_underrun | Output | underrun status flag for s_axis_srx AXI4 bus |

## Interfaces

The DPD IP core requires various interfaces to user logic for successful operation of the design. In addition to clocks and general control/sideband signalling the DPD v5.0 core has adopted AXI4-Stream and AXI4-Lite interfaces for both data and control respectively. The presence of AXI4-Lite and AXI4-Stream interfaces brings standardization and enhanced interoperability of Xilinx IP LogiCORE solutions. For further details on AXI4 interfaces see [Ref 3], [Ref 4] and [Ref 5].

### Clock Interface

This core requires up to four clock input signals. All of these clocks can be asynchronous to each other, allowing the phase and frequency of each to be independent.

#### *aclk*

Signal `aclk` is the primary clock used to drive all the datapath logic within the design. Its clock frequency is CLOCKS_PER_SAMPLE times the input data rate of the DPD IP core. The input data rate is also the sample rate at the output of DPD. All AXI4-Stream interfaces are synchronous to this clock.

#### *proc_clk*

Signal `proc_clk` is the processor clock used to drive all logic within the MicroBlaze subsystem. Increasing the speed of this clock reduces the time involved in the software aspects of the update algorithm. The `gp_input/gp_output` signals are directly connected to the MicroBlaze subsystem and are therefore synchronous to this clock.

#### *accel_clk*

Signal `accel_clk` is available when HWA is enabled. There is no particular minimum frequency to run this signal at; however, the higher the clock frequency is, the faster the computations are performed. Running at the same rate as the `proc_clk` should give acceptable acceleration. The effect of acceleration is calibrated in the SW Features Timing Performance section. This clock does not drive any external interface logic and is completely internal to the design.

#### *s_axi_aclk*

Signal `s_axi_aclk` is a dedicated input for the AXI4-Lite interface which provides asynchronous access to the shared memory for passing instructions to, and reading status from the DPD IP MicroBlaze processor.

The clocks can be generated using a combination of DCMs, PLLs and BUFGs, but care has to be taken to ensure that jitter specifications are used while constraining the clock signals. The actual frequency at which these clocks can run eventually depends upon other user logic, placement and timing constraints. For stand-alone characterization of typical clock frequencies, see IP Timing Performance.

The IP core does not contain any global clock resources like global buffers (`BUFGs`) and input global buffers (`IBUFG/IBUFGDS`). When the user instantiates the design, the clocks should be driven from a global clock buffer (any of the variants of `BUFGCTRL`) external to the IP core. Where the user chooses to let the synthesis tool infer these resources, care must be taken to ensure they are present. When the user instantiates the DPD netlist as a black box the synthesis tool might not recognize the clock ports and does not therefore infer a clock buffer. Also, if clocks are forwarded to the DPD netlist through another black box netlist in the users design then cascaded buffers can be created. Failing to ensure the correct clock resources are present can cause excessive skew on the clock distribution and various timing violations. Moreover, care needs to be taken to ensure that cascaded BUFGs aren't present in the design, due to clock forwarding through black box netlist. When a BUFG drives another BUFG using local routing instead of global clock routing resources, excessive clock skews can occur. This can cause various setup and hold violations in static timing analysis.

It is recommended that only stable clocks be connected to the IP when using DCM and PLL to derive the clocks. In that case, `BUFGCE` (or `BUFGMUX`) can be used, and lock signal can be used to derive the enable signal to connect to `BUFGCE CE` port.

### Reset Interface

The two reset signals `aresetn` and `s_axi_aresetn` have active low sensitivity. In the design, `aresetn` is the main reset and should be provided synchronous to `aclk`; this is used to reset all logic and interfaces in the design apart from the AXI4-Lite interface which has it own dedicated reset, `s_axi_aresetn`. Internally `aresetn` is re-synchronised to `proc_clk` and `accel_clk` to reset logic within these domains. The other reset,

s_axi_aresetn should be provided synchronous to s_axi_aclk and resets all AXI4-Lite interface logic and consequently prevents any pending block RAM accesses or responses.

Internal registers aid the routing of aresetn throughout the design while minimizing fanout. The MAX_FANOUT attribute is applied to the internal version of the reset signals, and the value of the attribute is set to REDUCE. This instructs the map tool to employ physical synthesis rules to reduce their fanout as per the requirements of the timing constraints. To assist in this, map should be run with -register_duplication turned on.

It is recommended that the user connect some form of reset to PLLs and DCMs generating the DPD design clock to ensure that there is a known start-up sequence, and ensure that the DCMs and PLLs have locked before de-activating the aresetn and s_axi_aresetn signals. If the clocks supplied to the DPD core become unstable while the PLLs are switching clock source (for example, during CPRI clock recovery), it is recommended that the DPD core be kept reset.

### Datapath AXI4-Stream Interfaces

Input and output data from the main datapath is supplied by means of slave and master AXI4-Stream interface respectively. This data comprises 32-bits for every transmit path present which is made up of a 16-bit real component in the lower 16 bits and a 16-bit imaginary component in the upper 16 bits. Based on an understanding of the signal statistics and dynamics, the user can pad or round data of a different bit widths to form the input data for a particular transmit path; where the bit width is less than 16 bits it should be zero padded at the LSB and where it is greater than 16 bits it should be symmetrically rounded. The data out contains a corresponding 32 bits for each transmit path which the user can connect to a digital mixer or complex DAC. If the TX DAC has a lower bitwidth (14 bits, for example), symmetric rounding should be applied to convert the signals from 16 to 14 bits. Symmetric rounding is required to prevent any unwanted DC offset, which introduces a spur in the transmitted spectrum in cases where the signal itself has no DC energy, for example WCDMA carrier configurations [0 0 1] and [1 0 1].

The core can be configured for a specific data rate; using the CLOCKS_PER_SAMPLE (CPS) parameter, a data rate of 1 to 4 clock-cycles/sample can be selected. The data rate is controlled at the AXI4-Stream interface through READY-VALID handshaking. After power-up or reset, the core is idle until S_AXIS_DIN_TVALID has been asserted for the first time; S_AXIS_DIN_TREADY is held high and M_AXIS_DOUT_TVALID is held low by the core until this occurs. After this, the core is synchronized and expects data input and provides data output every CPS clock cycles based on the configured data rate. S_AXIS_DIN_TVALID can be asserted at the same time as S_AXIS_DIN_TREADY, asserted in advance, or held asserted between S_AXIS_DIN_TREADY assertions. Note where the data rate is set to 1 clock per sample S_AXIS_DIN_TVALID is expected to be asserted permanently.

After the first time S_AXIS_DIN_TVALID is asserted, if S_AXIS_DIN_TVALID is not subsequently asserted when S_AXIS_DIN_TREADY is held high by the core, then the underrun condition occurs because valid data was not available when the core expected it. As the core is free-running, underrun causes the preceding data sample to be input for a second time internally. Underrun is signalled by asserting the EVENT_S_DIN_UNDERRUN flag for a single clock cycle after the condition has occurred.

After the core is synchronized by the slave interface, it periodically asserts M_AXIS_DOUT_TVALID to indicate valid output samples on the port M_AXIS_DOUT_TDATA. Similarly, the data rate defines how frequently new data is output and therefore how many cycles are available to signal that this data has been read. The core expects the data to be read by asserting M_AXIS_DOUT_TREADY before the next output data sample is available. Output data can be read immediately or on a later cycle but must be read prior to the next output data becoming available. After M_AXIS_DOUT_TVALID has been asserted to indicate the sample has been read, M_AXIS_DOUT_TVALID is deasserted unless the next output sample is available on the following clock cycle; in this case M_AXIS_DOUT_TVALID remains asserted while M_AXIS_DOUT_TDATA is updated with new output data. In the case of a data rate of 1 cycle per sample M_AXIS_DOUT_TVALID remains asserted and

M_AXIS_DOUT_TREADY is expected to be driven the same way while the data updates with a new output sample every cycle.

If an output sample is not read by asserting M_AXIS_DOUT_TREADY before the next is available, then the overrun condition occurs. As the core is free-running, overrun causes the next output sample to be dropped internally, as the current one has not been read and remains on the output port M_AXIS_DOUT_TDATA. Overrun is signalled by asserting the EVENT_M_DOUT_OVERRUN flag for a single cycle.

Figure 11 shows an example timing diagram where data-rate = 1 clock-cycle/sample and Figure 12 shows an example timing diagram where data rate = 3 clock-cycles/sample. The timing diagrams show the first slave valid pulse which synchronizes the interfaces and initiates the free running flow of data into and out of the core. The diagrams show different ready/valid handshaking scenarios on both slave and master including demonstrating invalid operation as signalled by the underrun and overrun flags.

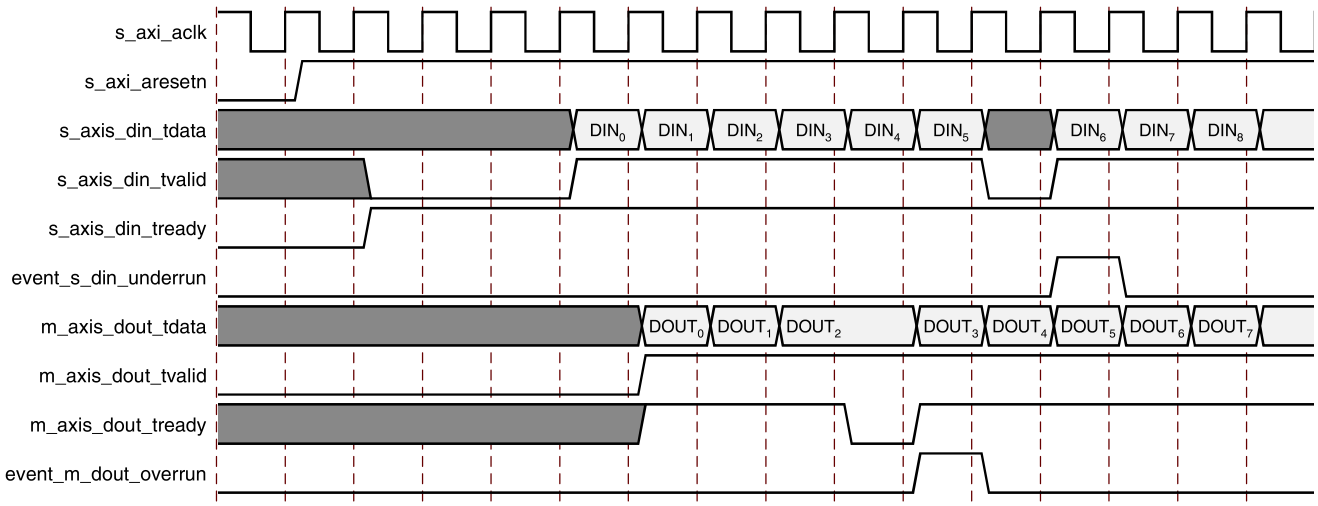


*Figure 11:* **AXI4-Stream Datapath Interface (CPS=1)**

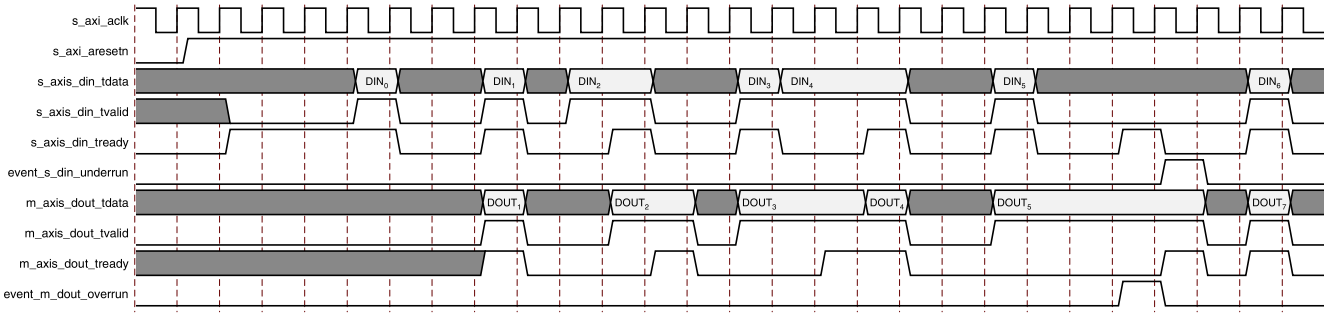


*Figure 12:* **AXI4-Stream Datapath Interface (CPS=3)**

### *Latency*

If the system integration requires an understanding of the latency through the core, refer to the end-to-end latency in data samples for various parameter combinations, listed in Table 2.

*Table 2:* **Latency per Transmit Antenna Path with no QMC or Rx_QMC (Independent of POLY_ORDER and HWA)**

| Performance Architecture | CLOCKS_PER_SAMPLE | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| A | 24 | 22 | 18 | 18 |
| B | 25 | 23 | 19 | 19 |
| C | 26 | 24 | 20 | 20 |
| D | 26 | 24 | 21 | 20 |

When QMC is set to Tx_QMC, latency per path increases by 4 data samples. The phase of the internal clock enables varies in relation to the input and output data phase which can add an additional sample to the latency.

## Sample Receiver (SRx) AXI4-Stream Interface

This AXI4-Stream slave interface operates in exactly the same way as the slave data input interface documented previously in Datapath AXI4-Stream Interfaces. See Figure 11 and Figure 12 for the relevant timing diagrams which demonstrate how this behaves on power up/reset and relative to the specified parameter CLOCKS_PER_SAMPLE. This interface operates independently of the main data path and can therefore receive its first valid pulse after power up/reset completely asynchronously to the main interface. The interface allows received (feedback) data to be input back into the core. The data should be presented in one of three formats determined by the RXINPUTFORMAT SW parameter value – 0, 1 or 2 (see Setting DPD Parameters and Sample Rates).

*Table 3:* **SRx Interface Signal Connection**

| RXINPUTFORMAT[107] | srx_din0 | srx_din1 | Notes |
|---|---|---|---|
| 0 ($2f_s$) | S0, S2, S4,… | S1, S3, S5,… | Combines to produce a data stream at rate $2f_s$ in which an srx_din0 datum should precede an srx_din1 datum. |
| 1 (IQ) | real | imag | The SPECTRALINVERSION SW parameter might need to be adjusted. |
| 2 ($f_s$) | 0 | ADC data | |

The $2f_s$ rate data generated from ADCs can be converted to `srx_din0/srx_din1` signals by either taking advantage of interleaving capabilities of various ADCs or, if the ADC is running at twice the rate, using ODDRs, a dual-aspect FIFO or a simple demux circuit with a FIFO to provide the `srx_din0` and `srx_din1` with appropriate data.

The $f_s$ rate data or complex IQ data from ADCs can be wired into the DPD ports using a simple asynchronous FIFO to transfer data from ADC clock domain to the `clk` domain.

### *Capture Synchronization*

In capture mode 1 (see Setting DPD Parameters section) the capture is triggered CAPTUREDELAY samples after a rising-edge is detected on the input signal `capture_sync` which is internally sampled by `aclk`. The user can assert `capture_sync` for one or more cycles of `aclk`; the capture is triggered relative to when it was asserted and it is irrelevant how many cycles it is asserted for.

## General Purpose I/O (gp_input / gp_output)

The signals `gp_input` and `gp_output` are directly connect to the processor memory space and are therefore synchronous to `proc_clk`. If a FIFO ADC interface is required, these signals are re-used for that purpose. Otherwise, the signal `gp_input` can be tied to ground and `gp_output` can be left unconnected. Details about FIFO ADC interactions are covered in FIFO ADC Interface.

### Host AXI4-Lite Interface

The Hardware Description section describes briefly the host interface and how it is tied into the MicroBlaze processor design. This interface allows the user to provide settings and instructions to the MicroBlaze processor about the various functions it needs to perform. This interface uses a 512x32 block RAM in dual-port mode. The AXI4-Lite slave interface gives full access to one port of this memory. See the AMBA AXI4 Interface Protocol web site for more information on the AXI4 and AXI4-Lite ports and operation.

The Operations Guide section provides the detail and usage of the memory map of this interface.

It should be noted that as there is only a single port of the block RAM available for accesses, write and read operations cannot take place simultaneously. The internal logic between the AXI4-Lite interface and the block RAM arbitrates, giving read accesses priority and therefore throttling write address/data, where required. If either write or read responses are not read (ready asserted) by the user then once approximately 8 responses of either operation have been buffered for output, the input of the corresponding operation is throttled by the core. It is possible to enter corresponding write address and write data out of sync; however, if full bandwidth (back to back access) is required these should be written simultaneously; otherwise either the address or data channel is throttled depending on which has been written already and which is still pending. When writing write address and write data out of sync the write bandwidth is limited to a maximum of 50%. As read accesses take priority; these always sustain full bandwidth at the expense of stalling write accesses.

### Multiple Antenna Interface

*Table 4:* **Bit Selection Indices for Multiple Antenna Interface**

| TX index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| din/dout_tdata indices | [31:0] | [63:32] | [95:64] | [127:96] | [159:128] | [191:160] | [223:192] | [255:224] |

For each TX data input, the Q component occupies the 16 MSBs and the I component occupies the 16 LSBs. For the SRx port, there are only 32 bits available where the 16 MSBs are identified by `srx_din0` and 16 LSBs as `srx_din1`. The MicroBlaze processor assumes the `srx_din` ports are being driven with data from path 0, 1, 2, 3, 4, 5, 6 or 7 when `srx_path_sel` is driven to 0, 1, 2, 3, 4, 5, 6 or 7 respectively. If the number of paths is set to 1, users can leave the `srx_path_sel` output port unconnected. If there are independent ADCs for all antennas, the switching of each TX paths feedback data onto the SRx port should be done in the FPGA using `srx_path_sel`. If the board has an RF switch to select between the output of the various PAs and then send the signal to a single ADC, `srx_path_sel` should be forwarded to that RF switch. If direct hardware connection to the RF switch or mux is not possible, the Antenna Selection Options in a Multipath Installation section should be consulted.

## Using Design Files

### Instantiate Netlist

Users can generate the core with appropriate parameters (see DPD GUI Parameters). According to the CORE Generator™ project settings, a netlist with vho/veo and vhd/v files are generated in the project directory with the file names matching the component name value. The user can instantiate the netlist file in their code following the template shown in the vho/veo file. Users can also use the vhd/v file for simulation purposes; these files are machine generated using UNISIM library components. The netlist and vhd/v files contain memory with pre-initialized software code to perform the pre-distortion functionality.

## Using Constraints

Users should take advantage of multicycle path constraints and cross-clock domain constraints to get better and faster static timing closure on their design. The DPD core is optimized to benefit from this. Users can add period constraints mentioned in Table 5 if the clocks are independently generated. The period constraints might get derived by the tools, if these clocks are generated from DCM/PLL/MMCM. The NET names used in this table should be updated with the appropriate signal name as connected to the DPD core's netlist clock inputs. Users should replace the string "<*CLK_PERIOD>" with appropriate period values in nanoseconds.

*Table 5:* **Example Period Constraints**

```
# FPGA System Clock Input
NET "aclk" TNM_NET = "TNM_clk";
TIMESPEC "TS_clk" = PERIOD "TNM_clk" <CLK_PERIOD> ns HIGH 50%;

# PROCESSOR CLOCK INPUT
NET "proc_clk" TNM_NET = "TNM_proc_clk";
TIMESPEC "TS_proc_clk" = PERIOD "TNM_proc_clk" <PROC_CLK_PERIOD> ns HIGH 50%;

# HWA Clock Input - Uncomment the next two lines, if HWA is selected
# NET "accel_clk" TNM_NET = "TNM_accel_clk";
# TIMESPEC "TS_accel_clk" = PERIOD "TNM_accel_clk" <ACCEL_CLK_PERIOD> ns HIGH 50%;

# HOST INTERFACE CLOCK INPUT
NET "s_axi_aclk" TNM_NET = "TNM_hi_clk";
TIMESPEC "TS_hi_clk" = PERIOD "TNM_hi_clk" <HI_CLK_PERIOD> ns HIGH 50%;
```

For internal multicycle paths, add the multicycle path constraint as shown in Table 6. Users should replace the string "<CEN_PERIOD>" with appropriate period values in nanoseconds. It is typically <CLK_PERIOD> value replaced in Table 5, multiplied by CLOCKS_PER_SAMPLE.

*Table 6:* **Example Multicycle Path Constraints**

```
# If instantiation is as follows at the topmost level of user code,

# dpd_inst : dpd_v4_0_component_name
# port map ( ….

#... );

# Multicycle path constraint (uncomment the next two lines, when CLOCKS_PER_SAMPLE=2,3 or 4)
# NET "dpd_inst/*grp1*" TNM_NET = "ce_N_group";
# TIMESPEC "TS_ce_N_group_to_ce_N_group" = FROM "ce_N_group" TO "ce_N_group" <CEN_PERIOD> ns;

#If the DPD instantiation is at a lower hierarchy, then update the NET name to reflect hierarchy.
```

Because the DPD IP uses up to four clocks and one multicycle path domain, it is useful to exploit the robustness in cross-clock domain crossing logic within the DPD design, by adding the cross-clock domain constraints as shown in Table 7. These rely on the timing groups defined in Table 5, so users should carefully replace the timing group names if they do not match in their environment. Apply caution when using TIG constraints and evaluate whether any of the user code (non dpd netlist) is getting covered by these timing groups and timespec constraints, and if so, verify that TIG constraints can be validly applied to the user logic as well. Consult Xilinx Support if you are concerned about the applicability of these constraints in your design.

*Table 7:* **Example Cross-clock Domain Constraints**

```
#CROSS CLOCK Domain TIG Constraints (uncomment the next 4 lines, if CLOCKS_PER_SAMPLE =1)
# TIMESPEC "TS_aclk_to_proc_clk_group" = FROM "TNM_clk" TO "TNM_proc_clk" TIG;
# TIMESPEC "TS_proc_clk_to_aclk" = FROM "TNM_proc_clk" TO "TNM_clk" TIG;
# TIMESPEC "TS_aclk_to_hi_clk_group" = FROM "TNM_clk" TO "TNM_hi_clk" TIG;
# TIMESPEC "TS_hi_clk_to_aclk" = FROM "TNM_hi_clk" TO "aclk" TIG;
#
#CROSS CLOCK Domain TIG Constraints (uncomment the next 4 lines, if CLOCKS_PER_SAMPLE =2,3 or 4)
# TIMESPEC "TS_ce_N_to_proc_clk_group" = FROM "ce_N_group" TO "TNM_proc_clk" TIG;
# TIMESPEC "TS_proc_clk_to_ce_N_group" = FROM "TNM_proc_clk" TO "ce_N_group" TIG;
# TIMESPEC "TS_ce_N_to_hi_clk_group" = FROM "ce_N_group" TO "TNM_hi_clk" TIG;
# TIMESPEC "TS_hi_clk_to_ce_N_group" = FROM "TNM_hi_clk" TO "ce_N_group" TIG;

#Uncomment the next two lines if HWA is enabled in the design
# TIMESPEC "TS_proc_clk_to_accel_clk_group" = FROM "TNM_proc_clk" TO "TNM_accel_clk" TIG;
# TIMESPEC "TS_accel_clk_to_proc_clk_group" = FROM "TNM_accel_clk" TO "TNM_proc_clk" TIG;
```

### Running Ngdbuild, Map and Par Tools

Regarding map tool options, it is recommended that the user set the following options in map.

- register_duplication on
- ignore_keep_hierarchy

The register_duplication switch is set to on and is essential to get the tools to manage the fanout on high fanout signals within the DPD IP (for example, clock enables and resets). The ignore_keep_hierarchy option in map is essential to allow better optimization for high fanout signals when they are used outside the DPD IP as well as improving the resource utilization of the logic across hierarchy. It is not required for the purpose of ignoring the hierarchy in the ngdbuild tool.

When running map and par, when timing closure becomes difficult for congested designs, it is recommended that the user see [Ref 6].

### Running Simulation

Users have the option to simulate the netlist generated, using the UNISIM based model provided during generation. Users should ensure that precompiled UNISIM libraries are available for the correct simulation tool version before proceeding to simulate. For the supported version of the tools, see the ISE Design Suite 13: Release Notes Guide.

A typical simulation test bench includes:

- Clock source generators
- Reset generators
- Data generators (AXI4-Stream input (din) and feedback (srx) data, sideband signal `capture_sync`)
- Data collectors (AXI4-Stream output (dout) data and sideband signal `srx_path_sel`)
- Host interface (AXI4-Lite write and read accesses)

Because the DPD netlist includes a processor and software code, simulation is slow. When QMC is not incorporated in the netlist, the DPD netlist exhibits pass through behavior after reset is released; the latency of the pass through is described in the Latency section. When QMC is enabled, the DPD netlist requires the internal MicroBlaze processor to initialize QMC; this introduces a long period between reset being released and the design exhibiting pass through behavior. During the initialization period, the DPD output might be undefined and it can take up to

25000 proc_clk cycles before input data is passed to the output with no modification. The actual number is lower for smaller values of TX and ARCH and when HWA is absent.

To exercise the host interface with various commands, the Operations Guide section should be referred to, for correct interaction procedure.

### Demonstration Test Bench

Contact Xilinx Technical Support for the Demonstration Test Bench.

## Migrating to DPD v5.0 from Earlier Versions

### XCO Parameter Changes

*Table 8:* **XCO Parameter Changes**

| XCO parameter name | Changes from v4.0 to v5.0 |
|---|---|
| component_name | No change. |
| clocks_per_sample | No change. |
| hardware_acceleration | false, true becomes 0,1 and two new values 2,3 available now. |
| number_of_transmit_antennas | No change. |
| polynomial_order | No change. |
| predistortion_correction_architecture | No change. |
| quadrature_modulation_corrector | false, true becomes None, Tx_QMC and new value Rx_QMC available now. |

### Port Changes

*Table 9:* **Port Changes from v4.0 to v5.0**

| DPD v4.0 Port | DPD v5.0 Port | Comments |
|---|---|---|
| clk | aclk | Clock and reset signals for input/output data and srx ports. Reset is now active low to match AXI4 requirements. ceN_out output signal is removed and the rate is managed by the AXI4 interface using tvalid/tready. Multicycle path constraints are used within the core and no external clock enables are offered. |
| rst | aresetn | |
| ceN_Out | - | |
| proc_clk | proc_clk | |
| proc_rst | - | A reset synchronized to proc_clk is now generated internally from aresetn. |
| accel_clk (optional) | accel_clk (optional) | Clock for any acceleration engine. |
| din_i/din_q/capture_sync | s_axis_din_*, capture_sync | Streaming real-time input data (16 bits I & 16-bits Q per transmit channel concatenated) now is described with AXI4-Stream slave interface and tready signal forces the data to be provided only at 1 every N cycles, where N is clocks per sample. capture_sync use remains as before. |
| gp_input, gp_output | gp_input, gp_output | These ports now used for FIFO ADC Interface. |
| dout_i/dout_q | m_axis_dout_* | Streaming real-time output data (16 bits I & 16-bits Q per transmit channel concatenated) now is described with AXI4-Stream slave interface. It is expected that tready should be toggled 1 in every N cycles, where N is clocks per sample period, so that output data is drained at sample rates. Otherwise, overrun status signal will be flagged. |
| srx_din0/srx_din1, srx_path_sel | s_axis_srx_*, srx_path_sel | Streaming real-time SRx data (16-bit srx_din0 - LSW concatenated with 16-bit srx_din1 - MSW) now is described with AXI4-Stream slave interface and tready signal forces the data to be provided only at 1 every N cycles. srx_path_sel output port use remains as before. |
| host_interface_* | s_axi_* | Host_interface (simple memory mapped) is replaced with AXI4-Lite interface. |
| - | event_* | Additional error events reported for various AXI4 interfaces. |

# Operations Guide

## Host Interface and SW Control Modes

DPD is controlled via the host interface RAM (see Host AXI4-Lite Interface). It is a port of a shared memory in the MicroBlaze processor subsystem that is typically connected to a microcontroller bus in the control plane of the application. DPD operations are activated by writing data to addresses in the RAM. Status, results, diagnostics and data are accessed by reading data from addresses.

The host interface memory map is organized into the regions as shown in Table 10. In what follows, individual addresses are specified. Addresses not explicitly referenced should be considered as reserved except for the range 216 to 255, which should be considered unused. For ease of reference and support, upper-case mnemonics are defined for key addresses, parameters, control modes and values. Where necessary, the associated value is shown in braces following the mnemonic.

*Table 10:* **DPD Host Interface Memory Map Outline**

| Address Range | Usage |
|---|---|
| 0 to 15 | Control |
| 16 to 95 | Continuous Monitors (read only) |
| 96 to 215 | Parameters (write only) |
| 256 to 383 | DCL Diagnostics (antenna 4 thru 7) (read only) |
| 384 to 511 | DCL Diagnostics (antenna 0 thru 3)/ Page Transfer (read only) |

DPD features are executed via triggering control modes (see Control Handshake). They are like function calls with optional parameters that influence the behavior of the datapath and internal state, in addition to returning results. Control modes are provided that allow the user to configure DPD, run single-step estimation, run the DCL and access measurements, read data and status information for setup, debug, and monitoring. Table 11 details the general registers involved with control modes.

*Table 11:* **DPD Host Interface Control Registers**

| Address | Mnemonic | Description |
|---|---|---|
| 0 | CONTROLMODETRIGGER | A control mode is triggered by toggling this register from zero to 0xABCDEF12 |
| 1 | CONTROLMODEREGISTER | The number of the control mode to execute |
| 2 | PORTNUM | Specify the port to which to apply command |
| 3 | WAITINGONPORTSWITCH | See Antenna Selection Options in a Multipath Installation |
| 8 | COMMANDSTATUS | See Table 13 |
| 9 | EXECUTEDCOMMAND | Echoes CONTROLMODEREGISTER on completion of the control mode |
| 10 | TRIGGERACK | Trigger Acknowledge |
| 11 | CODEPOINTER | See Control Handshake or Antenna Selection Options in a Multipath Installation |
| 12 | ACTIVEPORT | See Antenna Selection Options in a Multipath Installation |
| 13 | EXECUTINGCOMMAND | Reports the currently executing command |

### Control Handshake

The control modes should be triggered using the CODEPOINTER register to facilitate handshaking between the DPD core and the host environment. The valid values for the CODEPOINTER register are detailed in Table 12.

*Table 12:* Code Pointer Values

| Value | Mnemonic | Description |
|---|---|---|
| 1 | COMMAND_IN_PROGRESS | The processor is busy executing a requested command |
| 128 | WAITING_FOR_TRIGGER | The code is ready to execute a new command |
| 129 | WAITING_FOR_TRIGGER_RESET | The current command has completed. When the trigger is reset the code is ready for the next command. |
| 130 | WAITING_FOR_TRIGGER_RESET_AT_START | Trigger needs to be reset before bootup can complete. |
| 131 | WAITING_FOR_PORT_SWITCH | (Multiple antenna installations only.) The code is sitting idle waiting for an acknowledgement that the RF port has been switched as requested. (See Antenna Selection Options in a Multipath Installation) |

The necessary sequence of events is as follows (see Figure 13):

1. Verify CODEPOINTER is WAITING_FOR_TRIGGER.
2. Write the number of the required control mode to CONTROLMODEREGISTER.
3. Write the port (antenna) number (0 to 7), if required, to PORTNUM.
4. Write optional parameters as specified in Setting DPD Parameters
5. Trigger the control mode via CONTROLMODETRIGGER as specified.
6. Wait for CODEPOINTER not equal to COMMAND_IN_PROGRESS (non-DCL modes only).
7. Read COMMANDSTATUS.
8. Reset CONTROLMODETRIGGER.

Excepting DCL modes, modes terminate and return a status value in COMMANDSTATUS. Possible returned values are given in Table 13, referenced to the control modes involved.

*Table 13:* **Status Values**

| Relevant Control Mode(s) | Value | Mnemonic | Description |
|---|---|---|---|
| ALL | -511 | EVAL_LICENSE_TIMEOUT | The evaluation license has timed out. |
| COMPUTE_NEW_COEFFICIENTS(2) | -400 | HWA_HANDSHAKE_TIMEOUT | Hardware accelerator handshake has timed out. |
| COMPUTE_NEW_COEFFICIENTS(2) | -399 | HWA_HANDSHAKE_TIMEOUT | Hardware accelerator handshake has timed out. |
| SET_QMC_PARAMS(26) | -255 | CONFIG_FAILURE_QMC_L | Invalid number of samples to process for QMC. |
| SET_QMC_PARAMS(26) | -254 | CONFIG_FAILURE_QMC_NE | Invalid number of updates to average for QMC. |
| UPDATE_ECF_PARAMETERS(17) | -126 | CONFIG_FAILURE_L | The number of samples to process is outside the valid range. |
| UPDATE_ECF_PARAMETERS(17) | -125 | CONFIG_FAILURE_DELAY | The minimum delay is greater than the maximum delay requested. |
| SET_METER_LENGTH(6) | -124 | CONFIG_METER_LENGTH | The meter length is out of the valid range. |
| UPDATE_ECF_PARAMETERS(17) | -123 | CONFIG_FAILURE_ARCH | Invalid architecture selection. |

*Table 13:* **Status Values *(Cont'd)***

| Relevant Control Mode(s) | Value | Mnemonic | Description |
|---|---|---|---|
| UPDATE_ECF_PARAMETERS(17) | -121 | CONFIG_FAILURE_DELAY_WIN_SIZE | The maximum delay search size is too big. |
| COMPUTE_NEW_COEFFICIENTS(2) | -120 | ALIGN_FAILURE | Signal alignment failure. |
| COMPUTE_NEW_COEFFICIENTS(2) | -119 | COEF_OVERFLOW | Internal coefficient overflow. |
| COMPUTE_NEW_COEFFICIENTS(2) | -115 | LEASTSQUARES_FAILURE | A numerical issue was encountered during coefficient estimation. |
| COMPUTE_NEW_COEFFICIENTS(2) | -114 | SCA_FAILURE | Failure to capture a statistically sufficient set of samples. |
| COMPUTE_NEW_COEFFICIENTS(2) | -113 | CAPTURE_FAILURE | Failure to capture new samples. |
| ALL | -112 | INVALID_PORT | Invalid port was selected. |
| GET_HISTOGRAM_PAGE[5], GET_CAPTURE_HISTOGRAM_PAGE[15] | -111 | HISTOGRAM_FAILURE | Histogram failure. |
| ALL | -1 | INVALID_COMMAND | An invalid command was requested. |
| ALL | 0 | ZERO | Undefined. |
| ALL | 1 | COMMAND_IN_PROGRESS | The processor is busy executing a requested command. |
| ALL | 2 | SUCCESSFUL | Successful completion of the requested command. |
| COMPUTE_NEW_COEFFICIENTS(2) | 255 | OVERDRIVE_DETECTED | Overdrive was detected for the current ECF update, the coefficients are NOT blocked. |
| COMPUTE_NEW_COEFFICIENTS(2) | 256 | OVERDRIVE_PROTECTED | Overdrive was detected for the current ECF update, the coefficients ARE blocked - not switched into the datapath. |

Figure 13 is a flow chart that shows the general control handshaking for DPD processes.

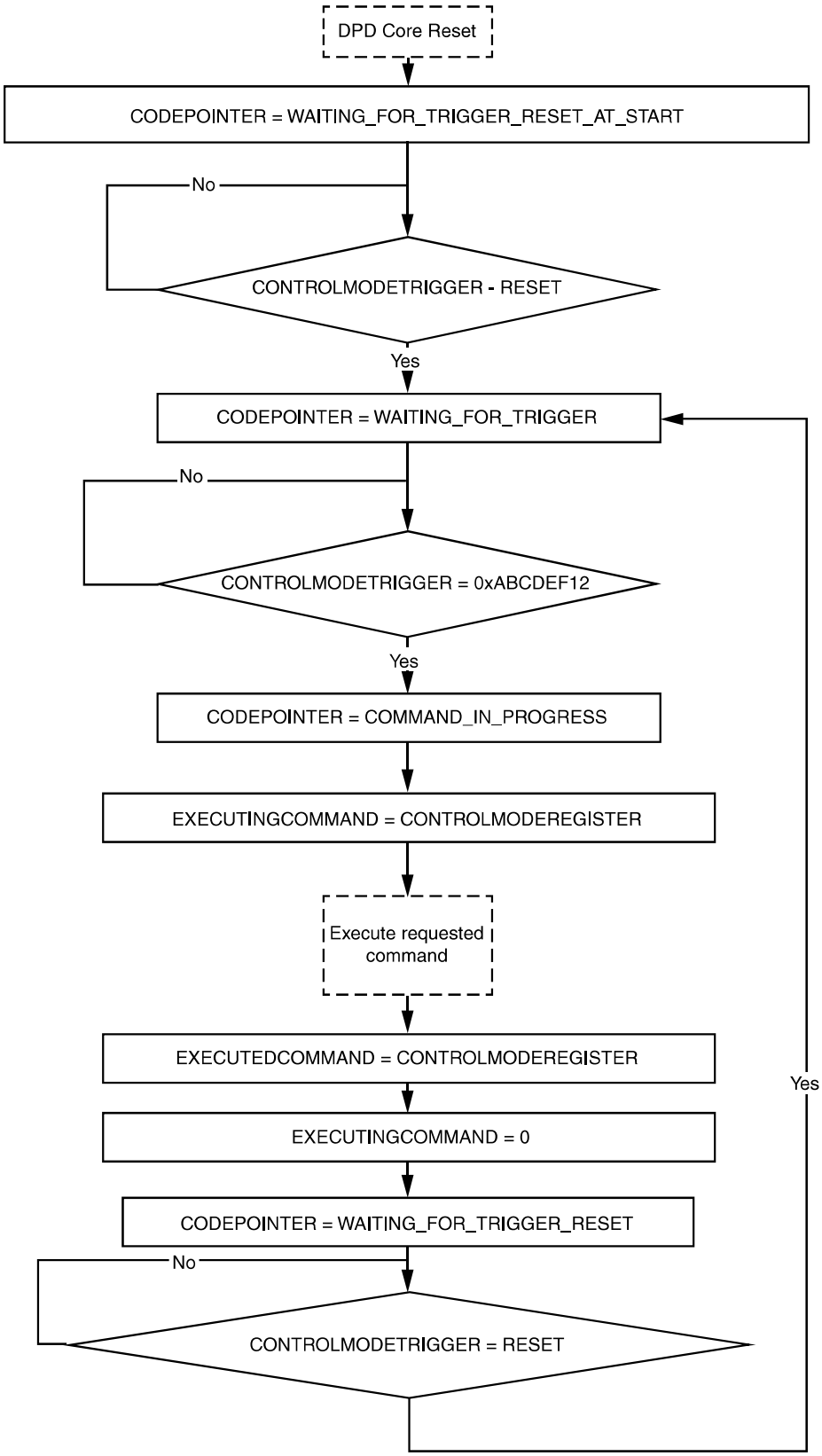


*Figure 13:* **Control Handshake**

## Setting DPD Parameters

The various parameters that control DPD are detailed in Table 14. Some parameters need to be set to suit the installation, particularly for the observation path configuration. For other parameters, the defaults usually provide good performance. Except where indicated, values are unsigned integers written to 32-bit registers. U32.x represent unsigned fractional values with x bits after the binary point.

*Table 14:* **DPD Parameters**

| Address | Mnemonic | Values | Default | Notes |
|---|---|---|---|---|
| 96 | SAMPLES2PROCESS | 400 to 4095 | 4000 | Number of samples to use for the DPD update algorithm; can be reduced to speed up estimation times (see SW Features Timing Performance) if performance is assured. |
| 99 | ARCH_SEL | (1) | | When D7 hardware is built there is a new software selectable architecture that is beneficial in wide band applications(2). |
| 102 | ODDTHRESHOLD | 0 to1 (U32.16) | 0.7 | Threshold to declare overdrive detection. |
| 103 | ODPENABLE | 0 or 1 | 0 | When ODP is enabled, coefficient sets that are predicted to cause an overdrive condition are not switched into the datapath. |
| 104 | DAMPEDNEWTONMU | 0 to 0.999 (U32.32) | 0.1 | Set to zero for unconditional Least-Squares updates. Set between 0 and 1 for damped updates. The DCL allows damped updates only when the power and loop gain is stable. Smaller non-zero values cause slower tracking with better suppressing of fluctuations at the cost of slower convergence. |
| 106 | SPECTRALINVERSION | 0 or 1 | 0 | When set to 1, this inverts the receive spectrum. Needed for low-side RF LO in the observation path. |
| 107 | RXINPUTFORMAT | 0 or 1 or 2 | 0 | 0 – the receiver is supplying real IF data at 2 x $f_s$.<br>1 – the receiver is supplying IQ baseband data at $f_s$.<br>2 – the receiver is supplying real IF data at $f_s$. |
| 108 | RXPHASESTEP | 0 to 0.5 (U32.32) | 0.25 (0x40000000) | (IF frequency/$f_s$) x $2^{32}$ for real IF receiver modes. |
| 109 + n | MINMAX_DELAY_ADJ | 200 ≥ (max-min) ≥ 32 | max 200, min 0 | Minimum and maximum integer sample delay to search over during initial delay adjustment. The maximum is specified in the 16 MSBs and the minimum is specified in the 16 LSBs. Here *n* ranges from 0 to the number of ports minus one. For example, when max is170 and min = 30 $1.200 \geq (170 - 30) \geq 32$ so the range is valid.<br>2. U32 value written is 00AA001E |
| 117 | CAPTUREMODE* | $M + L \times 2^3 + S \times 2^4$ | 0 | M - Capture mode.<br>0 – use SCA<br>1 – capture at fixed delay from supplied capture_sync signal<br><br>L - Capture location<br>0 – internal rx capture RAM<br>1 – external FIFO ADC<br><br>S - Pulse stretching<br>{0 ... 7} Amount to stretch the I/O read/write of the buffered ADC interface<br>(see FIFO ADC Interface, page 31) |

*Table 14:* **DPD Parameters *(Cont'd)***

| Address | Mnemonic | Values | Default | Notes |
|---|---|---|---|---|
| 118 | CAPTUREDELAY* | 0 to $2^{32}$-1 | 15000 | Capture delay in samples at $f_s$ associated with capture mode 1. |
| 119 | METERLENGTH** | $2^{18}$ to $2^{24}$-1 | 1228800 | Number of samples for measurements block processing. Should be a multiple of any frame size and $\geq 10ms$. |
| 123 | QMCSAMPLESTO PROCESS**** | 400 - 4095 | 2000 | Number of samples to use for the QMC update algorithm. |
| 124 | QMCNUMCAPTURES**** | 1 - 256 | 16 | QMC updates to average prior to updating the registers. Effectively a multiplier to the QMCSAMPLESTOPROCESS parameter. |
| 125 | QMC_GAIN_MU**** | U32.24 | 1.0 | Scaling term applied to the gain error prior to updating the register. |
| 126 | QMC_PHASE_MU**** | U32.24 | 1.0 | Scaling term applied to the phase error prior to updating the register. |
| 127 | QMC_OFFSET_MU**** | U32.24 | 1.0 | Scaling term applied to the offset errors prior to updating. |
| 128 | DCLPSTEP*** | DCL Parameters | 255 | Control the decay rate for the reference levels. |
| 129 | DCLSET1RATIO*** | U16.16 (DCL Parameters) | 0.8 | Set threshold for second coefficients set relative to the first set. |
| 130 | DCLSET2RATIO*** | U16.16 (DCL Parameters) | 0.5 | Set threshold for third coefficients set relative to the first set. |
| 134 | DCL_MODE*** | 0 or 1 | 0 | 0 – single set mode<br>1 – multiple set mode<br>(Mode 1 now supports the use of the DAMPEDNEWTONMU parameter. For v4.x performance set DAMPEDNEWTONMU parameter to zero.) |

**Notes:**

1. When the core is instantiated, the performance architecture GUI selection sets the hardware provision for the pre-distortion function. By default, the ECF recognizes this and computes the appropriate number of coefficients. However, for evaluation purposes the ECF can use a lesser degree than provisioned in the core. ARCH_SEL in Table 14 can be set using the following relationship to enable non-default configurations.

$$\text{ARCH\_SEL} = x + (y - 5) \times 2^3$$

   where x is 1, 2, 3 or 4 representing ARCH = A, B, C and D respectively and y is the polynomial value POLY_ORDER = 5 or 7.
   For example D/7th order is ARCH_SEL =20 and C/5th order is ARCH_SEL =3.
   The higher values of `ARCH_SEL` are normally beneficial only for signals where the occupied bandwidth is greater than around one sixth to one seventh of the pre-distortion bandwidth $f_s$, and `ARCH_SEL` should be set to the minimum value required to achieve best performance. See Sample Rates.
2. When D7 hardware is built there is a new software selectable architecture (D7f) that is beneficial for wide band applications. D7f is enabled by setting ARCH_SEL = 84.The D7f architecture is more complex than D7 and has long update times (>10 seconds) when no hardware accelerators are used. D7f is typically not beneficial for narrow band signals. The use of D7f with narrow band signals can cause an excessive number of COEF_OVERFLOW errors to occur.

Parameters can be written into the host interface RAM at any time, but are not activated until a control mode is executed. For unmarked items the command mode is UPDATE_ECF_PARAMETERS(17);
otherwise the command mode is:

* - SET_CAPTURE_PARAMS(11),** - SET_METER_LENGTH(6), *** -SET_DCL_PARAMETERS(12) - **** - SET_QMC_PARAMS(26).

## DCL Parameters

The performance of the DCL can be tuned using four parameters. The DCLPSTEP, DCLSET1RATIO, and DCLSET2RATIO are set using the SET_DCL_PARAMETERS(12) control and the METERLENGTH which is set using the SET_METER_LENGTH(6) control. The meter length is used to set the interval over which the power is measured and is the update period for coefficient set selection when multiple set DCL is used.

The DCLSET1RATIO and DCLSET2RATIO are used to set the thresholds that determine which coefficient set should be used. These thresholds are set relative to the maximum power that is to be transmitted.

The DCLPSTEP is used to set a decay rate for the three thresholds which ensures system gain tracking over time.

Figure 14 shows an example of the DCL behavior with the default multiple set parameters. In this example, the power is slowly ramped up and down by 6 dB. The tx power is read back from the internal power meter and displayed in the figure.

- The three solid thin lines are the threshold values used by the DCL
- The color coded small dots indicate which coefficient set is currently loaded into the DPD filter
  - The active coefficient in the DPD filter is updated at the interval defined by METERLENGTH.
- The larger color coded dots indicate when and at what power level each coefficient set is being updated.
  - A coefficient set is updated whenever the current power is above the decayed threshold (color coded dashed lines).

At start-up, all three thresholds are zero and the power begins to ramp up. During the initial ramp up, the $P_{max}$ threshold (blue line) tracks the signal power. As the power begins to drop, $P_{max}$ is held and the decay begins to increment (dashed blue line). As the power drops below DCLSET1RATIO x $P_{max}$ and DCLSET2RATIO x $P_{max}$, the second and third thresholds are set, respectively, and their thresholds begin to decay. The set thresholds are again updated as the power rises and crosses the decayed threshold values. The blue, green, and red color coding of the power measurement curve indicates which coefficient set the DCL has selected for use (set0, set1, and set2, respectively).

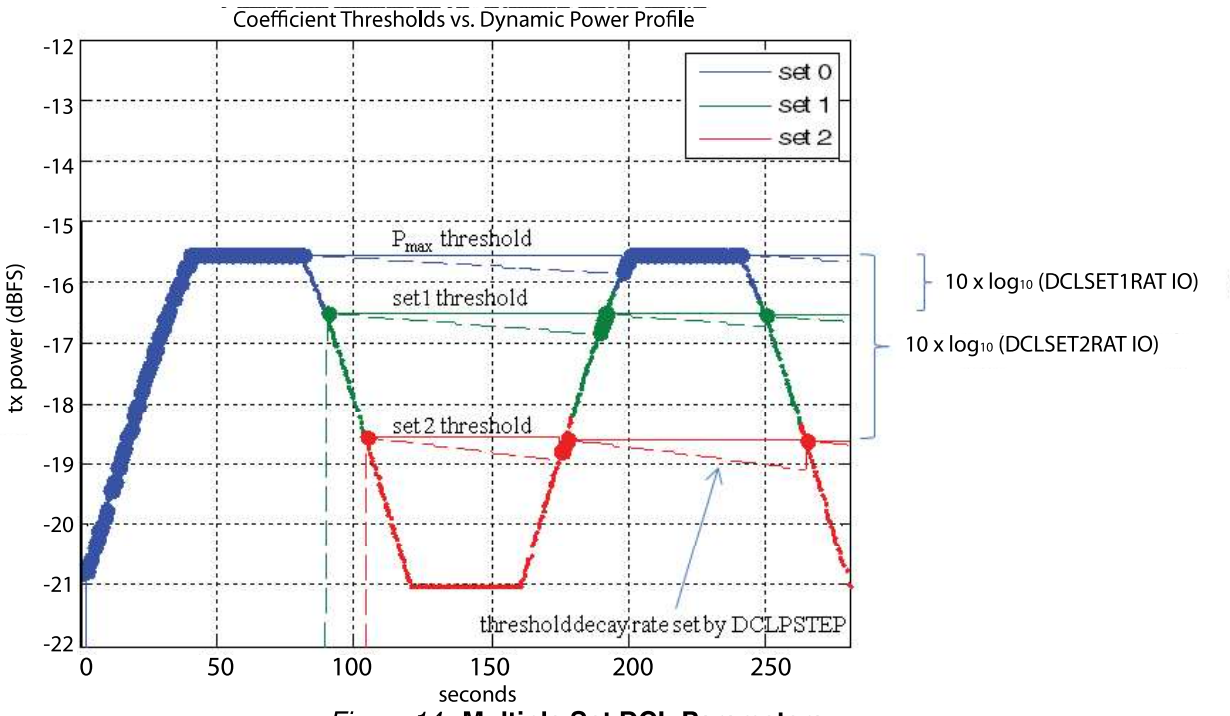


*Figure 14:* **Multiple Set DCL Parameters**

### Determining DCLPSTEP Parameter

The parameter DCLPSTEP controls the rate of decay for the reference levels. To obtain a required rate, the following procedure should be used:

The decay rate (DR) in linear power units is

$$DR_{linear} = \left\lfloor \frac{DCLSTEP \times F_s}{2^{28}} \right\rfloor$$

Where $F_s$ is the MicroBlaze clock frequency in Hz. A convenient method for selecting DCLSTEP is to set its value such that the $P_{max}$ threshold decays by 3 dB over some defined time ($\Delta t$) in seconds.

$$DCLSTEP = \left\lfloor \frac{2^{49} \times 10^{\left(\frac{P_{max}}{10}\right)}}{\Delta t \times F_s} \right\rfloor$$

An example is shown here where the expected $P_{max}$ is set to -15 dBFS and the threshold is set to decay by 3 dB in 10 minutes with a clock rate of 122.88 MHz

$P_{max}$ = -15 dBFS
$\Delta t$ = 600 s
$F_s$ = 122.88 MHz

$$DCLSTEP = \left\lfloor \frac{2^{49} \times 10^{\left(\frac{-15}{10}\right)}}{600 \times 122.88 \times 10^{6}} \right\rfloor = 247$$

## FIFO ADC Interface

This interface can be used to transfer `SRx` samples from an external FIFO using the general purpose I/O's. The interface can be used with a general class of burst mode ADC devices (referred to here as a FIFO ADC) with just a small amount of external resource. The `gp_input` and `gp_output` ports are interfaced to the ADC via some supporting logic and a 12 bit by 8192 deep FIFO. The DPD SW trigger echoes its internal capture trigger to the ADC, the ADC then dumps to the internal FIFO and then DPD copies the data into its internal capture RAM.

### Logic Design and Operation

The `gp_input` and `gp_output` ports are customized to support access to FIFO ADC's data. These ports are synchronous to `proc_clk` clock domain and any logic reading or writing to these ports is better managed when operating using `proc_clk`. It is useful to keep cross-clock domain signals local to user design, so that constraints do not cross over DPD's I/O ports.

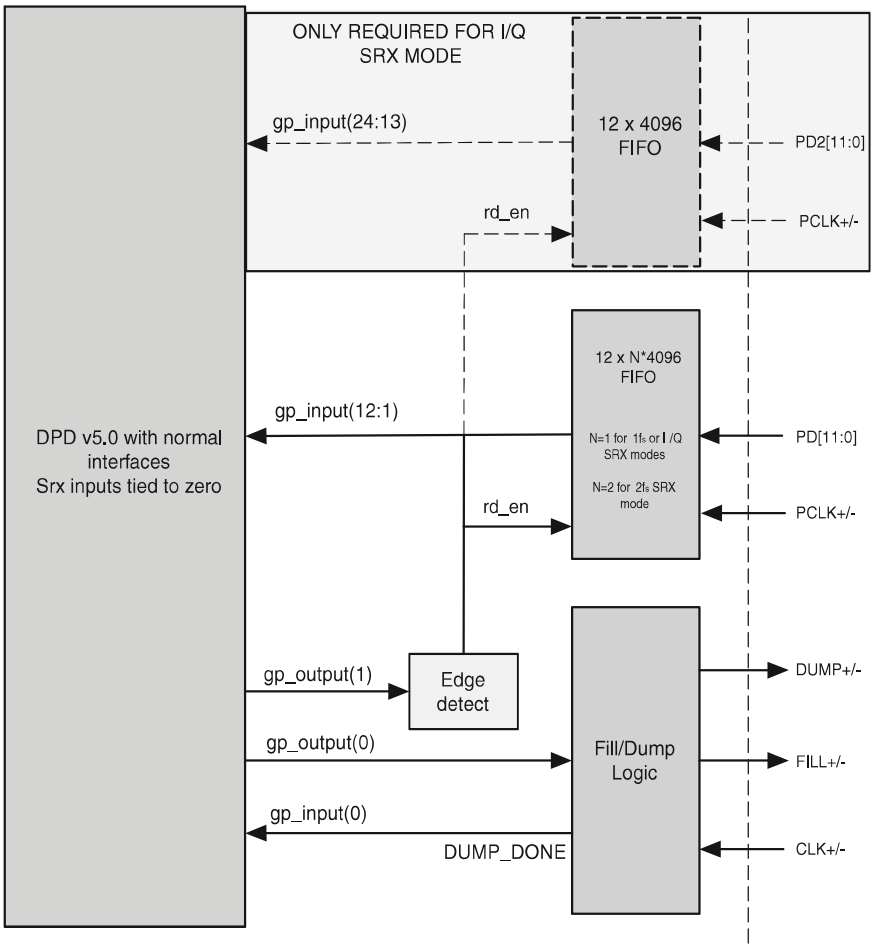


*Figure 15:* **Block Diagram**

### *Fill/Dump Logic operation*

The FIFO ADC data sheet should be consulted - the signals FILL and DUMP are synchronous to the main ADC clock input which needs to be made available to the FPGA. When DPD performs a capture, a rising edge appears on the `gp_ouput(0)` pin. When this occurs, the Fill/Dump logic block should initiate a fill (single pulse on the FILL input of the ADC(s)), wait for an appropriate time, and then assert the DUMP signal for a sufficient time to fill the FIFO(s). When this is complete, the DUMP_DONE signal should display a positive edge transition, which is interpreted by DPD that the data is ready for use. There are a number of software cycles between the assertion of `gp_output(0)` and the time when DPD begins monitoring `gp_input(0)`, so it should be possible for the hardware to use the edge detection of the assertion of `gp_output(0)` to reset `gp_input(0)`.

### *FIFO*

The FIFO(s) is (are) filled by the ADC in the PCLK domain. The read side needs to be clocked by `proc_clk` clock signal. The `gp_output(1)` signal should be positive edge detected in this domain to provide the read enable for the FIFO(s). The sequence of these acts to copy the data from the FIFO(s) into DPD's internal capture RAM.

***Note:*** The software only copies the samples from the FIFO(s) into the capture RAM after successfully passing the SCA criteria. In the event that SCA_FAILURE occurs during the capture attempt, the rx samples in the capture RAM do not correspond to the latest capture attempt.

### *SRx mode support*

The FIFO ADC interface supports all three SRx modes supported by DPD v5.0. The real SRx modes read in 8192 or 4096 12-bit values from the `gp_input` depending on whether the 2fs or 1fs SRx mode is selected. The I/Q SRx mode reads in 4096 24-bit values from the `gp_input`. In I/Q SRx mode the ordering of the two 12-bit I/Q inputs is not important because they can be swapped in software using the spectral inversion command.

### Timing interaction

The capture process starts when DPD asserts the `gp_output(0)` bit and then begins monitoring `gp_input(0)`. After the DPD core detects an assertion on `gp_input(0)` the process of copying the samples from the FIFO into DPD begins. DPD asserts `gp_output(1)` for a number of cycles (the length of the pulse can be extended as part of the capture parameters to ensure the hardware has enough time to detect the edge). After DPD deasserts `gp_output(1)` it begins to read `gp_input(12:1)`, or `gp_input(24:1)` if I/Q mode is used (the amount of time spent reading is controlled by the same sample capture parameter, to ensure the hardware has enough time to get the next sample from the FIFO.)

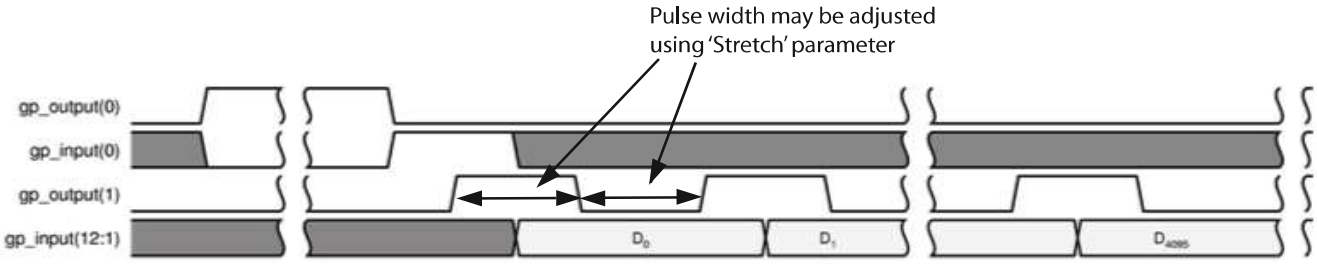


*Figure 16:* **Timing Interaction**

## Monitors

Various information is always available from the host interface RAM. This is detailed in Table 15.

*Table 15:* **Monitors**

| Address | Mnemonic | Description |
|---|---|---|
| 34 | VERSION_0 | Information to be supplied when contacting Xilinx Support. |
| 35 | VERSION_1 | |
| 48 | RUNTIMELSW | 32-bit LSW of a time monitor that counts the number of measurement intervals. The timer is reset when the meter length is updated or when the DCL parameters are set. |
| 49 | RUNTIMEMSW | MSW of the time monitor. |
| 50 | SRXLSW | 32-bit LSW of the receiver power. |
| 51 | SRXMSW | 32-bit MSW of the receiver power. |
| 52 + 5n | TXPOWERLSW | 32-bit LSW of the transmit power of port n (where it ranges from 0 to number of ports minus one). |
| 53 + 5n | TXPOWERMSW | 32-bit MSW of the transmit power of port n (where it ranges from 0 to number of ports minus one). |
| 54 + 5n | TXPOWERCOUNT | Number of non-zero samples in the power measurement of port n (where it ranges from 0 to number of ports minus one). |

Each power is a 64-bit value representing the sum of the individual powers of the number of samples specified in METERLENGTH, divided by 256. To convert to dBFS, the formula is:

$$10 \times \log_{10}(256 \times \text{power}/(2^{30} \times \text{TXPOWERCOUNT}).$$

## Running the DCL

Normally DPD is activated by using the DCL control modes (Table 16). PORTNUM need not be specified, as the DCL automatically operates on all available ports. Various status monitors are provided (Table 17) and can be used to implement error handling. In a multiport installation, n ranges from 0 to the number of ports minus one. To maintain backward compatibility with the v4.x DCL monitors, the location of the individual port monitors is defined in Table 16, where k is defined as (using C notation): $k = (n < 4 \ ? \ n : n - 8)$

In a system where DPD has been running successfully, the appearance of an undesired status in, for instance, UPDATE_INPROGRESS or LAST_UPDATED_STATUS typically indicates an abnormal signal condition such as a failed observation receiver or the occurrence of severe interference. If the ECF does encounter an error, the coefficients are not updated or stored. All ECF parameters are relevant to the DCL.

*Table 16:* **DCL Control Modes**

| Mode Number | Mnemonic | Description |
|---|---|---|
| 14 | RUN_DCL[14] | Run the DCL. |
| 18 | EXIT_DCL[18] | Stop the DCL while retaining internal state. |
| 23 | RUN_DCL_WITH_ACCEL_QMC[23] | Run the DCL with QMC updates enabled and QMC initial convergence is emphasized. See QMC support in DCL. |
| 27 | RUN_DCL_WITH_QMC[27] | Run the DCL with QMC updates enabled. See QMC support in DCL. |
| 36 | RESET_DCL[36] | Reset the DCL internal state.<br>Executing SET_DCL_PARAMETERS(12) also does this. |

*Table 17:* **DCL Monitors**

| Address | Mnemonic | Description |
|---|---|---|
| 392 + 32*k | COUNTER | The total number of ECF updates since the function was started. This is the last register written in this table. Detecting a change in this register can be used to indicate all values in this table are stable for reading, subject to the beginning of the next update. |
| 395 + 32*k | UPDATE_INPROGRESS | 0 – no ECF updates are currently occurring<br>1 – an ECF update using Damped-Newton is being computed<br>2 – an ECF update using Least-Squares is being computed<br>32 – TX power is too low to compute ECF updates<br>33 – RX power is too low to compute ECF updates |
| 396 + 32*k | LAST_TIME_MSW | 32-bit MSW of the time stamp corresponding to the last update. Counted in MicroBlaze clock cycles. |
| 397 + 32*k | LAST_TIME_LSW | 32-bit LSW of the time stamp corresponding to the last update. |
| 399 + 32*k | LAST_UPDATED_STATUS | Indicates the returned status of the ECF update code (see Table 13). |
| 400 + 32*k | QMC_GAIN_ERROR | Use 20 x $\log_{10}$(QMC_GAIN_ERROR) to convert to dB error (U32.16). |
| 401 + 32*k | QMC_PHASE_ERROR | Use QMC_PHASE_ERROR x 180/$\pi$ to convert to error in degrees (S32.29). |
| 402 + 32*k | QMC_I_OFFSET_ERROR | LSB DC offset (S32.16). |
| 403 + 32*k | QMC_Q_OFFSET_ERROR | LSB DC offset (S32.16). |
| 404 + 32*k | LAST_QMC_UPDATED_STATUS | Indicates the returned status of the QMC update code. |
| 405 + 32*k | QMC_UPDATE_COUNTER | The total number of QMC updates since the function was started. |
| 406 + 32*k | MEASURED_RX_POWER | Average SRx power; when FIFO ADC interface is being used this is the average power in the rx capture RAM. |
| 409 + 32*k | ODDMETRIC | ODD metric (U32.16). |
| 410 + 32*k | ODDPEAKEXPANSION | Estimated peak expansion of the waveform.<br>Use 20 x $\log_{10}$(ODDPEAKEXPANSION) to convert to dB (S32.16). |

**Note:** In the event of contacting Xilinx Support, it is useful to have a record of the DCL monitors.

## QMC support in DCL

The Dynamic Control Layer (DCL) supports two modes of operation. These modes differ only in their start-up performance. The steady state performance of both modes is the same, the ECF and QMC updates are interleaved (1 QMC capture for every ECF update). Note that one QMC Update = QMCNUMCAPTURES QMC captures. This behavior is shown in Figure 17 and illustrates the start-up performance (using default parameters) when using the RUN_DCL_WITH_QMC(27) command. In this case, QMCNUMCAPTURES is set to 16, so while the ECF has made 80 updates the QMC has made only 5.

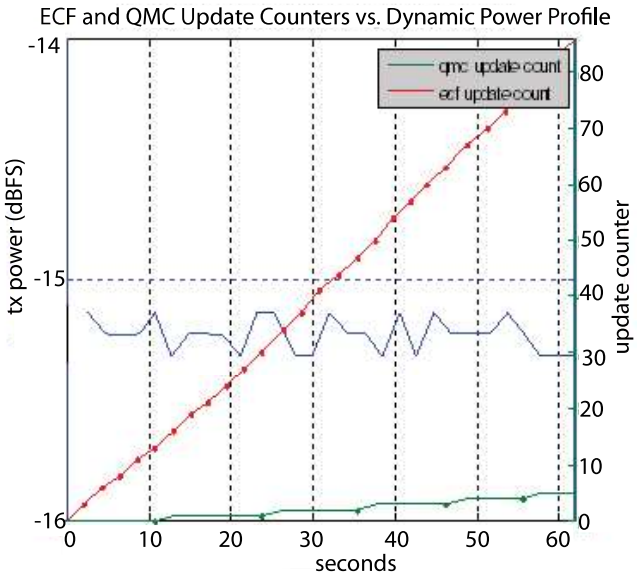


*Figure 17:* **Startup performance using the RUN_DCL_WITH_QMC{27} command**

This might be acceptable when the initial QM error is small (for example, starting with pre-calibrated/corrected QMC coefficients), but might be undesirable when starting with a larger QM error. For this case the RUN_DCL_WITH_ACCEL_QMC(23) command can be used to emphasize the QMC updates during the initial seconds of operation. In this mode, the QMCNUMCAPTURES is temporarily set to 1 and the ECF vs. QMC updates is defined using following pseudo code:

```
// accelerated QMC updates
<set QMCNUMCAPTURES = 1>
for (n = 10; n > 0; n--) {
    <perform ECF update>
        for(m=0; m<n; m++)
            <perform QMC update>
}
// normal QMC updates
<set QMCNUMCAPTURES = user defined parameter>
```

This gives 55 QMC updates (using QMCNUMCAPTURES = 1) during the first 10 ECF updates. After this initial phase the ECF vs. QMC update interleaving is the same as the method used with the RUN_DCL_WITH_QMC(27) command. This behavior is shown in Figure 18 and illustrates the start-up performance (using default parameters) when using the RUN_DCL_WITH_ACCEL_QMC(23) command.

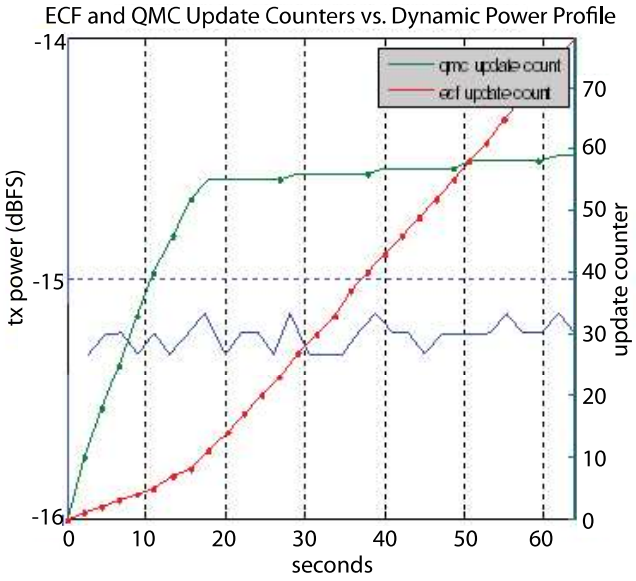


*Figure 18:* **Startup performance using the RUN_DCL_WITH_ACCEL_QMC{23} command**

## Single Stepping

For fine control, parameter adjustment, debug, general understanding and non-standard applications DPD software features can be individually activated via the control modes given in Table 18. PORTNUM(2) needs to be specified.

*Table 18:* **Single Stepping Control Modes**

| Mode Number | Mnemonic | Description |
|---|---|---|
| 1 | READ_CONFIGURATION | Refresh the host interface with internal parameters. |
| 2 | COMPUTE_NEW_COEFFICIENTS | Perform a full Least-Squares update of the DPD coefficients. This includes capturing new samples, ECF processing and updating the datapath parameters. |
| 3 | RESET_COEFFICIENTS | Set all coefficient sets to unity gain and update the datapath for pass-through. |
| 7 | DPD_OFF | Update the datapath for pass-through without changing the internally stored coefficients. |
| 8 | DPD_ON | Update the datapath from the internally stored coefficients. Used for single set DCL and single iteration commands. |
| 9 | DPD_ON_1 | Update the datapath from the internally stored coefficients. Used for multiple set DCL. |
| 10 | DPD_ON_2 | Update the datapath from the internally stored coefficients. Used for multiple set DCL. |
| 19 | RESTORE_DEFAULTS | Restore the default configuration. |
| 21 | RESET_QMC | Reset the internally stored QMC coefficients and set the QMC datapath block to pass-through. |
| 22 | QMC_SINGLE_STEP | Update QMC by making a single iteration. |
| 24 | QMC_ON | Update the QMC datapath block from the internally stored coefficients. |
| 25 | QMC_OFF | Update the QMC datapath block for pass-through without changing the internally stored coefficients. |
| 33 | DAMPED_UPDATE | Perform a Damped-Newton iteration of the DPD coefficients. This includes capturing new samples, processing the ECF, and updating the datapath parameters. |

## Tuning the QMC parameters

### Quadrature Modulator (QM) Error Detectors

Equation 6 thru Equation 9 are computed in software for the detection of QM errors. A mixture of fixed-point and double precision are used to keep the code fast. In the equations that follow the TX and RX signals are the captured samples after signal alignment. The gain imbalance is computed by measuring the ratio of power between the I and Q channels of the TX samples vs. the ratio of I and Q powers in the RX samples.

$$Gain_{imbalance} = \sqrt{\frac{\sum|Re(tx)|^2 \times \sum|Im(rx)|^2}{\sum|Im(tx)|^2 \times \sum|Re(tx)|^2}} \quad \text{Equation 6}$$

The DC offset errors are computed as the mean of the difference between the TX and RX samples.

$$I_{DC} = \frac{\sum[(Re(rx) - Re(tx))]}{L} \quad \text{Equation 7}$$

$$Q_{DC} = \frac{\sum[(Im(rx) - Im(tx))]}{L} \quad \text{Equation 8}$$

And the phase imbalance is detected by correlating the error between the TX and RX signals to the orthogonal signal component.

$$Phase_{imbalance} = \frac{\sum[Re(rx) - Re(tx) \times Im(tx)]}{\sum Im(tx) \times Im(tx)} + \frac{\sum[Im(rx) - Im(tx) \times Re(tx)]}{\sum Re(tx) \times Re(tx)} \quad \text{Equation 9}$$

The value L in Equation 6 thru Equation 9 represents the number of samples used to estimate the various QM error terms. The parameter L (see Determining QMC Parameters) can be used to tune the tracking error by controlling the amount of estimation error in the detectors. Increasing L reduces the estimation error, thereby reducing the tracking noise in the loop.

L also affects the tracking bandwidth of the loops because the time required to capture and compute the errors is proportional to the value L takes. The relation to actual tracking bandwidth is difficult to determine because the collection of the L samples is not always accomplished in a single step (see Determining QMC Parameters) and the actual update interval is not strictly deterministic (see Determining QMC Parameters). Knowledge of the actual tracking bandwidth is of less concern because QM errors vary extremely slowly over temperature and time; the primary concern in QMC is the tracking error.

Figure 19 shows an example of the effect L has on tracking error. The results shown in Figure 19 were collected while transmitting a single 10MHz LTE signal with a power amplifier in the RF path.

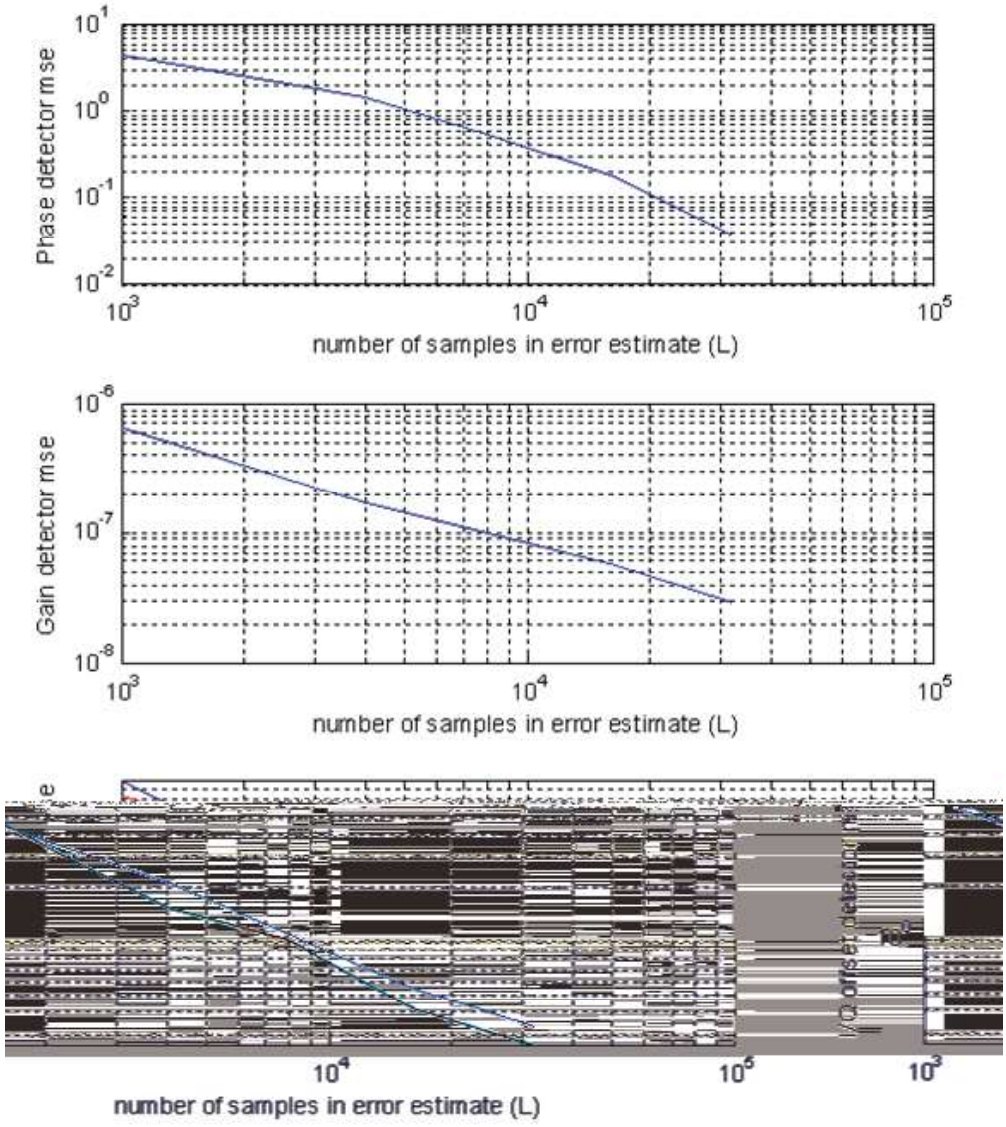


*Figure 19:* **Effect of Number of Samples (L) on Tracking Error**

## Updating the QMC Registers

The QMC registers are updated independently using first-order control loops (proportional loop) as seen in Equation 10 thru Equation 13.

$$qmc_{gain}(i+1) = qmc_{gain}(i) \times (1.0 + \mu_{gain}(Gain_{imbalance} - 1.0)) \quad \text{Equation 10}$$

$$qmc_{phase}(i+1) = qmc_{phase}(i) - \mu_{phase}(Phase_{imbalance}) \quad \text{Equation 11}$$

$$qmc_{i\_offset}(i+1) = qmc_{i\_offset}(i) - \mu_{offset}(I_{DC}) \quad \text{Equation 12}$$

$$qmc_{q\_offset}(i+1) = qmc_{q\_offset}(i) - \mu_{offset}(Q_{DC}) \quad \text{Equation 13}$$

Tracking bandwidth is always traded against tracking error when using first-order loops. Tracking bandwidth is of little concern in QMC because QM errors vary extremely slowly, so the primary concern is in achieving an acceptable low tracking error.

Figure 20 shows an example of the effect the parameters (see Determining QMC Parameters) have on the convergence of the loops.

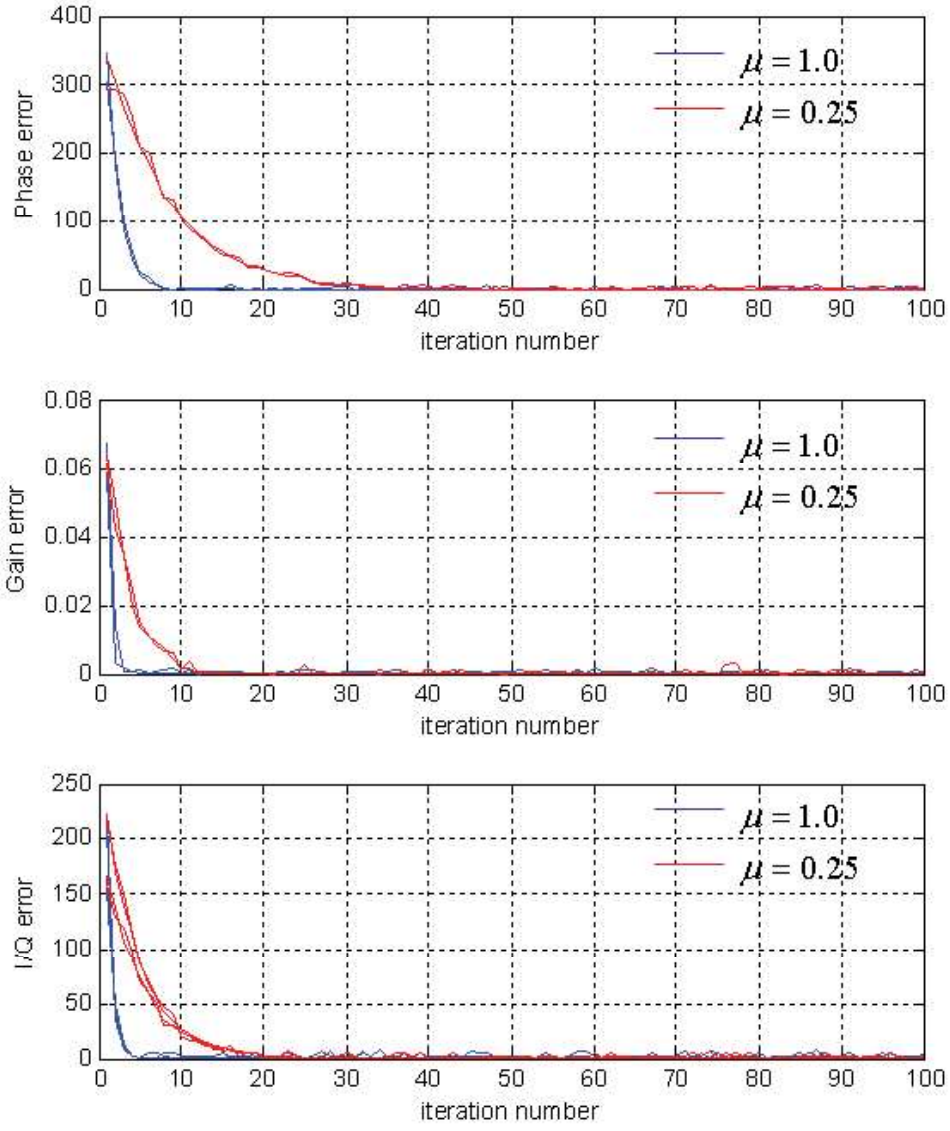


*Figure 20:* **Effect of Number of Gain Terms on Convergence Time**

## Determining QMC Parameters

It is expected that the default configuration of QMC is sufficient for most applications; however, the QMC algorithm does have five parameters that can be used to tune the performance for a given installation (see Table 14, page 28). The QMCSAMPLESTOPROCESS parameter instructs the MicroBlaze processor on how many samples to capture and process during each capture not each update. The capture RAM is limited to 4096 complex samples, but better QMC performance can be achieved by estimating the QM errors over a much greater number of samples. The

samples used for estimating QMC errors do not need to be strictly consecutive samples; hence multiple captures can be used to increase L in the update. The parameter QMCNUMCAPTURES determines how many captures to process before updating the QMC registers. The value L in Equation 6 thru Equation 9 is determined by multiplying QMCSAMPLESTOPROCESS and QMCNUMCAPTURES.

The time required to process a single capture (given a 122.88 MHz MicroBlaze processor clock with no hardware acceleration) is shown in Figure 21. The default for QMCSAMPLESTOPROCESS is set to 2000 samples (it was 1000 in v2.x thru v4.x) to reduce the impact of QMC updates on the DCL routine. QMCNUMCAPTURES is set to 16 (it was 32 in v2.x thru v4.x) by default, which makes L for the QMC update 32,000 samples. With these defaults each QMC update requires approximately eight seconds of processing time (assuming no alignment acceleration). That eight second duration is divided into 0.25 second intervals that are distributed intelligently in time by the DCL algorithm, thereby avoiding an otherwise 8 second window when DPD updates would not be available to the DCL algorithm.

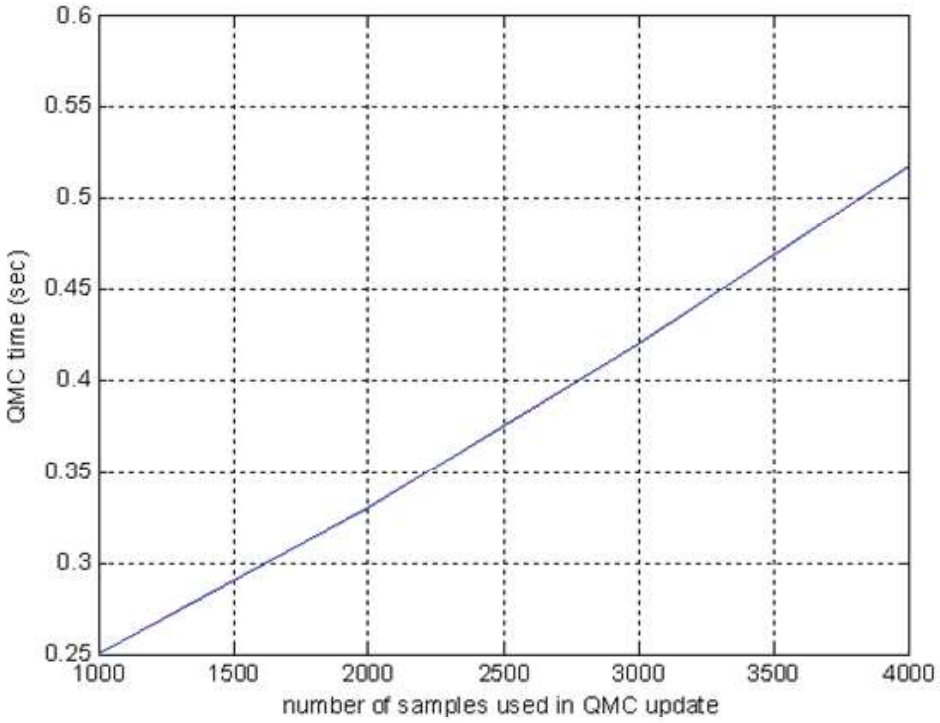


*Figure 21:* **QMC Processing Time vs. L**

Finally, all of the gain terms (QMC_GAIN_MU, QMC_PHASE_MU and QMC_OFFSET_MU) are defaulted to 1.0.

## Signal Analysis

To aid setup and debug, the control modes shown in Table 19 give access to the signals processed by DPD and measurements made by DPD on those signals. PORTNUM(2) needs to be specified.

A capture can be triggered and the captured data, the (transmit) power and histogram can be read out.

Transfer of bulk data uses a paged mechanism. Each page is 128 data long and is available at addresses 384-511.

The capture RAM is 8192 samples long and therefore requires 64 page accesses. The parameter PAGENUMBER (address 122) specifies a page number from 0 to 63. The first 4096 samples are the transmit data – each 32-bit data consists of a concatenation of two 16-bit twos-complement data for the I (LSW) and Q (MSW) samples. The upper

4096 samples are the receive data that are again a concatenation of two 16-bit twos-complement data. When the receiver is in 2*$f_s$ mode, these are the even and odd ordered samples of an 8192 sample sequence. In 1*$f_s$ mode, the odd samples are ignored, and in IQ mode the format is to pack the two 16-bit SRx inputs into a single 32-bit value. In all modes the SRx data appears in captured samples as `srx_din0` x $2^{16}$ + `srx_din1`. Table 3 provides details on `srx_din0/srx_din1` interface.

The histogram integrated over METERLENGTH can also be read out. The histograms are 256 samples long. The histogram bins are the number of samples of signal amplitude when amplitude is divided by 128.

The capture power is the sum of the capture signal power over the number of samples specified in SAMPLES2PROCESS(96).

*Table 19:* **Signal Analysis Control Modes**

| Mode Number | Mnemonic | Description |
|---|---|---|
| 4 | GET_CAPTURE_RAM_PAGE | Present the page, specified by PAGENUMBER(122), of the capture RAM data at the host interface RAM page transfer area. |
| 5 | GET_HISTOGRAM_PAGE | Present the page, specified by PAGENUMBER(122), of the transmit histogram at the host interface RAM page transfer area. |
| 15 | GET_CAPTURE_HISTOGRAM_PAGE | Present the page, specified by PAGENUMBER(122), of the capture histogram at the host interface RAM page transfer area. |
| 16 | READ_CAPTURE_POWER_METERS | Present the values from the capture TX power meter at addresses 384(LSW) and 385(MSW) and the capture RX power at addresses 386(LSW) and 387(MSW). |
| 20 | CAPTURE_NEW_SAMPLES | Trigger a new sample capture sequence. The capture follows the rules set by the CAPTUREMODE(117) parameter. |
| 37 | RESIDUAL_ERRORS | Perform capture and align with TX samples from before the DPD filter. |
| 38 | AMAM_AMPM_ERRORS | Perform capture and align with TX samples from after the DPD filter. |

Examples of uses for the signal analysis control modes are to:

- check the transmit and receive spectra (by analyzing the captured data in a tool such as MATLAB) and thereby verify that the signal source, RF paths, core interfaces and relevant DPD parameters are correct.
- check the CFR configuration (by examining the transmit histogram).
- determine appropriate settings for CAPTUREDELAY(118) in capture mode 1 by examining the capture histogram and powers relative to the measurements over METERLENGTH.

***Note:*** In the event of contacting Xilinx Support, it is useful to have the signal analysis data described in this section. Xilinx offers a fully featured MATLAB based GUI debug environment that can be used to quickly evaluate and explore the full capabilities of the DPD solution. Contact Xilinx Support for access to the debug interface environment [Ref 2].

## Reading and Loading Coefficient Set

The DPD and QMC coefficients can be read from the DPD core or loaded into the DPD core for initialization purposes. These processes can be executed only when DCL is not operational; hence the page transfer/DCL monitor area in host interface is used to transfer the coefficients.

### DPD coefficient handling

There are three coefficient sets (per transmit path) used when DCL_MODE is set to multiple set and a single coefficient set used when DCL_MODE is set to single set mode. When DCL_MODE is single set the only active coefficient set for each transmit path is set 0.

The coefficients values can only be accessed while DCL is disabled.

#### *Reading DPD coefficients sets*

1. Write required port to read coefficients from to address PORTNUM(2) in the host interface.
2. Write required coefficient set number to address PARAM_0(4). There are three sets available (0, 1 or 2).
3. Trigger DPD command REPORT_COEFFICIENTS(30).
4. Read values 380 thru 511 (132 values) from the host interface. These values are the coefficient set represented in a proprietary format.

#### *Loading DPD coefficient sets*

1. Write required port to write coefficients to in address PORTNUM(2).
2. Write required coefficient set number to address PARAM_0(4). There are three sets available (0, 1 or 2).
3. Write the 132 values read during the "Reading DPD coefficients sets" process into registers 380 thru 511.
4. Trigger DPD command LOAD_COEFFICIENTS(31). This command loads the coefficients into the micro-blaze memory, but does NOT load them into the DPD filter.
5. (Optional) Trigger command DPD_ON(8), DPD_ON_1(9) or DPD_ON_2(10) to load the required coefficient set into the DPD filter.

### QMC coefficient handling

The coefficients values can only be accessed while DCL is disabled.

#### *Reading QMC coefficients sets*

1. Write required port to read coefficients from to address PORTNUM(2) in the host interface.
2. Trigger DPD command REPORT_QMC_COEFFICIENTS(34)
3. Read values 384 thru 387 (4 values) from the host interface. These values are the coefficient set represented in a proprietary format.

#### *Loading QMC coefficient sets*

1. Write required port to write coefficients to in address PORTNUM(2).
2. Write the 4 values read during the "Reading QMC coefficients sets" process into registers 384 thru 387.
3. Trigger DPD command LOAD_QMC_COEFFICIENTS(35). This command loads the coefficients into the micro-blaze memory, but does NOT load them into the QMC registers.
4. (Optional) Trigger command QMC_ON(24) to load the required coefficient set into the QMC registers.

## Antenna Selection Options in a Multipath Installation

For multiple-antenna applications, DPD assumes that there is an RF or digital switch selecting the various observation paths to route to the sample receiver. The most transparent mode of operation is if the switch control is available as signals in the FPGA, in which case they should be wired to `srx_path_sel` port of the core.

In the event that the switch is accessible only via the application control plane, a SW handshake protocol is provided for switching the receiver. This is enabled by executing the ENABLE_EXT_RXSEL(28) control mode and disabled by ENABLE_INT_RXSEL(29).

To use this external select mode, these steps are to be followed:

1. Poll CODEPOINTER(11) until the value 131 is read. This is the request for a port switch.
2. Read the ACTIVEPORT(12) register to see which port needs to be switched in to the receive path.
3. Switch the port and acknowledge by writing 0xA5A5A5A5 into the WAITINGONPORTSWITCH(3) register.

In the internal mode, ACTIVEPORT indicates which port is active, and WAITINGONPORTSWITCH is ignored.

## Host Interface Memory Map

*Table 20:* **Memory Map**

| Word Address | Mnemonics | Notes |
|---|---|---|
| 0 | CONTROLMODETRIGGER | A control mode is triggered by toggling this register from zero to 0xABCDEF12. |
| 1 | CONTROLMODEREGISTER | The number of the control mode to execute. |
| 2 | PORTNUM | Specify the port to which to apply command. |
| 3 | WAITINGONPORTSWITCH | See Antenna Selection Options in a Multipath Installation. |
| 4 | reserved | |
| 5 | unused | |
| 6 | unused | |
| 7 | unused | |
| 8 | COMMANDSTATUS | See Table 13. |
| 9 | EXECUTEDCOMMAND | Echoes CONTROLMODEREGISTER on completion of the control mode. |
| 10 | TRIGGERACK | Trigger acknowledge. |
| 11 | CODEPOINTER | See Control Handshake or Antenna Selection Options in a Multipath Installation. |
| 12 | ACTIVEPORT | See Antenna Selection Options in a Multipath Installation. |
| 13 | EXECUTINGCOMMAND | Reports the currently executing command. |
| 14 | unused | |
| 15 | unused | |
| 16 | unused | |
| 17 | unused | |
| 18 | unused | |
| 19 | unused | |
| 20 | unused | |
| 21 | unused | |
| 22 | unused | |
| 23 | unused | |
| 24 | reserved | used for file versioning information. |
| 25 | reserved | used for file versioning information. |
| 26 | reserved | used for file versioning information. |
| 27 | reserved | used for file versioning information. |
| 28 | reserved | used for file versioning information. |
| 29 | reserved | used for file versioning information. |
| 30 | reserved | used for file versioning information. |
| 31 | reserved | used for file versioning information. |
| 32 | reserved | used for file versioning information. |
| 33 | reserved | DPD hardware settings register. |
| 34 | VERSION_0 | Build version information. |
| 35 | VERSION_1 | Build version information. |

*Table 20:* **Memory Map *(Cont'd)***

| Word Address | Mnemonics | Notes |
|---|---|---|
| 36 | reserved | |
| 37 | reserved | |
| 38 | reserved | |
| 39 | reserved | |
| 40 | reserved | |
| 41 | reserved | |
| 42 | unused | |
| 43 | unused | |
| 44 | unused | |
| 45 | unused | |
| 46 | unused | |
| 47 | reserved | |
| 48 | RUNTIMELSW | 32-bit LSW of a time monitor that counts the number of measurement intervals. The timer is reset when the meter length is updated or when the DCL parameters are set. |
| 49 | RUNTIMEMSW | 32-bit MSW of a time monitor that counts the number of measurement intervals. The timer is reset when the meter length is updated or when the DCL parameters are set. |
| 50 | SRXLSW | 32-bit LSW of the receiver power. |
| 51 | SRXMSW | 32-bit MSW of the receiver power. |
| 52 | TXPOWERLSW_A | 32-bit LSW of the transmit power of port n (where it ranges from 0 to number of ports minus one). |
| 53 | TXPOWERMSW_A | 32-bit MSW of the transmit power of port n (where it ranges from 0 to number of ports minus one). |
| 54 | TXPOWERCOUNT_A | Number of non-zero samples in the power measurement of port n (where it ranges from 0 to number of ports minus one). |
| 55 | reserved | |
| 56 | ACTIVESET_A | Status indicating which coefficient set is loaded into the predistortion LUTs. Can be used for monitoring during DCL operation. |
| 57 | TXPOWERLSW_B | 32-bit LSW of the transmit power of port n (where it ranges from 0 to number of ports minus one). |
| 58 | TXPOWERMSW_B | 32-bit MSW of the transmit power of port n (where it ranges from 0 to number of ports minus one). |
| 59 | TXPOWERCOUNT_B | Number of non-zero samples in the power measurement of port n (where it ranges from 0 to number of ports minus one). |
| 60 | reserved | |
| 61 | ACTIVESET_B | Status indicating which coefficient set is loaded into the predistortion LUTs. Can be used for monitoring during DCL operation. |
| 62 | TXPOWERLSW_C | 32-bit LSW of the transmit power of port n (where it ranges from 0 to number of ports minus one). |
| 63 | TXPOWERMSW_C | 32-bit MSW of the transmit power of port n (where it ranges from 0 to number of ports minus one). |

*Table 20:* **Memory Map *(Cont'd)***

| Word Address | Mnemonics | Notes |
|---|---|---|
| 64 | TXPOWERCOUNT_C | Number of non-zero samples in the power measurement of port n (where it ranges from 0 to number of ports minus one). |
| 65 | reserved | |
| 66 | ACTIVESET_C | Status indicating which coefficient set is loaded into the predistortion LUTs. Can be used for monitoring during DCL operation. |
| 67 | TXPOWERLSW_D | 32-bit LSW of the transmit power of port n (where it ranges from 0 to number of ports minus one). |
| 68 | TXPOWERMSW_D | 32-bit MSW of the transmit power of port n (where it ranges from 0 to number of ports minus one). |
| 69 | TXPOWERCOUNT_D | Number of non-zero samples in the power measurement of port n (where it ranges from 0 to number of ports minus one). |
| 70 | reserved | |
| 71 | ACTIVESET_D | Status indicating which coefficient set is loaded into the predistortion LUTs. Can be used for monitoring during DCL operation. |
| 72 | TXPOWERLSW_E | 32-bit LSW of the transmit power of port n (where it ranges from 0 to number of ports minus one). |
| 73 | TXPOWERMSW_E | 32-bit MSW of the transmit power of port n (where it ranges from 0 to number of ports minus one). |
| 74 | TXPOWERCOUNT_E | Number of non-zero samples in the power measurement of port n (where it ranges from 0 to number of ports minus one). |
| 75 | reserved | |
| 76 | ACTIVESET_E | Status indicating which coefficient set is loaded into the predistortion LUTs. Can be used for monitoring during DCL operation. |
| 77 | TXPOWERLSW_F | 32-bit LSW of the transmit power of port n (where it ranges from 0 to number of ports minus one). |
| 78 | TXPOWERMSW_F | 32-bit MSW of the transmit power of port n (where it ranges from 0 to number of ports minus one). |
| 79 | TXPOWERCOUNT_F | Number of non-zero samples in the power measurement of port n (where it ranges from 0 to number of ports minus one). |
| 80 | reserved | |
| 81 | ACTIVESET_F | Status indicating which coefficient set is loaded into the predistortion LUTs. Can be used for monitoring during DCL operation. |
| 82 | TXPOWERLSW_G | 32-bit LSW of the transmit power of port n (where it ranges from 0 to number of ports minus one). |
| 83 | TXPOWERMSW_G | 32-bit MSW of the transmit power of port n (where it ranges from 0 to number of ports minus one). |
| 84 | TXPOWERCOUNT_G | Number of non-zero samples in the power measurement of port n (where it ranges from 0 to number of ports minus one). |
| 85 | reserved | |
| 86 | ACTIVESET_G | Status indicating which coefficient set is loaded into the predistortion LUTs. Can be used for monitoring during DCL operation. |
| 87 | TXPOWERLSW_H | 32-bit LSW of the transmit power of port n (where it ranges from 0 to number of ports minus one). |
| 88 | TXPOWERMSW_H | 32-bit MSW of the transmit power of port n (where it ranges from 0 to number of ports minus one). |

*Table 20:* **Memory Map *(Cont'd)***

| Word Address | Mnemonics | Notes |
|---|---|---|
| 89 | TXPOWERCOUNT_H | Number of non-zero samples in the power measurement of port n (where it ranges from 0 to number of ports minus one). |
| 90 | reserved | |
| 91 | ACTIVESET_H | Status indicating which coefficient set is loaded into the predistortion LUTs. Can be used for monitoring during DCL operation. |
| 92 | unused | |
| 93 | unused | |
| 94 | unused | |
| 95 | unused | |
| 96 | SAMPLES2PROCESS | Number of samples to use for the DPD update algorithm; can be reduced to speed up estimation times (see SW Features Timing Performance) if performance is assured. |
| 97 | reserved | |
| 98 | reserved | |
| 99 | ARCH_SEL | Selected DPD architecture. |
| 100 | reserved | |
| 101 | reserved | |
| 102 | ODDTHRESHOLD | Threshold to declare overdrive detection. |
| 103 | ODPENABLE | When ODP is enabled, coefficient sets that are predicted to cause an overdrive condition are not switched into the datapath. |
| 104 | DAMPEDNEWTONMU | Set to zero for unconditional Least-Squares updates. Set between 0 and 1 for damped updates. The DCL allows damped updates only when the power and loop gain is stable. Smaller non-zero values cause slower tracking with better suppressing of fluctuations at the cost of slower convergence. |
| 105 | reserved | |
| 106 | SPECTRALINVERSION | When set to 1, this inverts the receive spectrum. Needed for low-side RF LO in the observation path. |
| 107 | RXINPUTFORMAT | 0 – the receiver is supplying real IF data at 2 x $f_s$.<br>1 – the receiver is supplying IQ baseband data at $f_s$.<br>2 – the receiver is supplying real IF data at $f_s$. |
| 108 | RXPHASESTEP | (IF frequency/$f_s$) x $2^{32}$ for real IF receiver modes. |
| 109 | MINMAX_DELAY_ADJ_A | Minimum and maximum integer sample delay to search over during initial delay adjustment. The maximum is specified in the 16 MSBs and the minimum is specified in the 16 LSBs. Here *n* ranges from 0 to the number of ports minus one.<br>For example, when max is 170 and min = 30<br>1. so the range is valid.<br>2. U32 value written is 00AA001E |
| 110 | MINMAX_DELAY_ADJ_B | |
| 111 | MINMAX_DELAY_ADJ_C | |
| 112 | MINMAX_DELAY_ADJ_D | |
| 113 | MINMAX_DELAY_ADJ_E | |
| 114 | MINMAX_DELAY_ADJ_F | |
| 115 | MINMAX_DELAY_ADJ_G | |

*Table 20:* **Memory Map *(Cont'd)***

| Word Address | Mnemonics | Notes |
|---|---|---|
| 116 | MINMAX_DELAY_ADJ_H | |
| 117 | CAPTUREMODE | M - Capture mode.<br>0 – use SCA<br>1 – capture at fixed delay from supplied capture_sync signal<br>L - Capture location<br>0 – internal rx capture RAM<br>1 – external FIFO ADC<br>S - Pulse stretching<br>{0 ... 7} Amount to stretch the I/O read/write of the buffered ADC interface (see FIFO ADC Interface, page 31) |
| 118 | CAPTUREDELAY | Capture delay in samples at fs associated with capture mode 1. |
| 119 | METERLENGTH | Number of samples for measurements block processing. Should be a multiple of any frame size and ≥ 10 ms. |
| 120 | reserved | |
| 121 | reserved | |
| 122 | PAGENUMBER | Page number to transfer. |
| 123 | QMCSAMPLESTOPROCESS | Number of samples to use for the QMC update algorithm. |
| 124 | QMCNUMCAPTURES | QMC updates to average prior to updating the registers. Effectively a multiplier to the QMCSAMPLESTOPROCESS parameter. |
| 125 | QMC_GAIN_MU | Scaling term applied to the gain error prior to updating the register. |
| 126 | QMC_PHASE_MU | Scaling term applied to the phase error prior to updating the register. |
| 127 | QMC_OFFSET_MU | Scaling term applied to the offset errors prior to updating. |
| 128 | DCLPSTEP | Control the decay rate for the reference levels. |
| 129 | DCLSET1RATIO | Set threshold for second coefficients set relative to the first set. |
| 130 | DCLSET2RATIO | Set threshold for third coefficient set relative to the first set. |
| 131 | reserved | |
| 132 | reserved | |
| 133 | reserved | |
| 134 | DCL_MODE | 0 – single set mode<br>1 – multiple set mode<br>(mode 1 now supports the use of the DAMPEDNEWTONMU parameter. For v4.x performance set DAMPEDNEWTONMU parameter to zero.) |
| 135-215 | Reserved for un-documented diagnostics. | |
| 216-255 | unused | |
| 256- 383 | DCL Diagnostics (antenna 4 thru 7) (read only) | |
| 384-511 | DCL Diagnostics (antenna 0 thru 3)/ Page Transfer (read only) | |

# Resource Utilization and Performance

## Resource Utilization

The DPD IP core provides various parameters and targets multiple FPGA families. Table 22 and Table 23 provide some examples of resource utilization on Virtex-6 and Kintex-7. These resource numbers were obtained when the designs were run stand-alone and all clock signals were constrained as described in Table 24.

ISE® 13.2 tools were used to obtain this data. Non-default settings used for the various programs are described in Table 21. These enable physical synthesis capability to offer better fanout management for high fanout internal multicycle paths and `aresetn`, `s_axi_aresetn` network. It is recommended that the user also employ these settings when instantiating DPD IP in user logic.

*Table 21:* **Tool Settings for Characterization**

| Tools | Additional Settings |
|---|---|
| XST | NONE |
| ngdbuild | NONE |
| map | -register_duplication on, -ignore_keep_hierarchy, -xe n |
| par | NONE |

*Table 22:* **Resource Utilization on Virtex-6**

| TX | Architecture | Clocks/ Sample | Poly. Order | QMC | HWA | FFs | LUTs | Slices | Block RAM 36K/18K(1) | DSP48E1s |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | D | 4 | 7 | 0 | 0 | 3927 | 3383 | 1752 | 55/0 | 14 |
| 1 | D | 4 | 7 | 1 | 0 | 4001 | 3375 | 1677 | 55/0 | 17 |
| 1 | D | 4 | 7 | 2 | 0 | 3884 | 3330 | 1679 | 55/0 | 14 |
| 1 | D | 4 | 7 | 1 | 1 | 5742 | 4533 | 2071 | 64/10 | 37 |
| 1 | D | 4 | 7 | 0 | 1 | 5644 | 4704 | 2072 | 64/10 | 34 |
| 1 | D | 4 | 7 | 0 | 2 | 6824 | 6097 | 2825 | 68/10 | 47 |
| 1 | D | 4 | 7 | 0 | 3 | 9162 | 7489 | 3655 | 68/10 | 78 |
| 2 | D | 4 | 7 | 0 | 3 | 10385 | 8611 | 3391 | 76/10 | 85 |
| 4 | D | 4 | 7 | 0 | 3 | 12989 | 9805 | 4793 | 92/10 | 99 |
| 8 | D | 4 | 7 | 0 | 3 | 18047 | 13470 | 6308 | 126/10 | 127 |
| 4 | D | 3 | 7 | 0 | 3 | 15599 | 11377 | 5334 | 124/10 | 115 |
| 4 | D | 2 | 7 | 0 | 3 | 15082 | 10841 | 5489 | 92/10 | 115 |
| 4 | D | 1 | 7 | 0 | 3 | 11972 | 10352 | 4190 | 124/10 | 147 |
| 2 | C | 4 | 7 | 0 | 3 | 10252 | 8301 | 3672 | 72/10 | 85 |
| 2 | B | 4 | 7 | 0 | 3 | 9929 | 7940 | 3907 | 68/10 | 85 |
| 2 | A | 4 | 7 | 0 | 3 | 9618 | 7809 | 3847 | 64/10 | 85 |
| 2 | A | 4 | 5 | 0 | 3 | 9585 | 7796 | 3834 | 64/10 | 85 |
| 2 | C | 4 | 5 | 0 | 0 | 4992 | 4047 | 2056 | 51/0 | 21 |
| 2 | C | 4 | 5 | 0 | 1 | 6771 | 5126 | 2790 | 68/10 | 41 |

**Notes:**

1. In some configurations Virtex-6 cases use a number of 18K block RAMs in addition to full 36K block RAMs.

*Table 23:* **Resource Utilization on Kintex-7**

| TX | Architecture | Clocks/ Sample | Poly. Order | QMC | HWA | FFs | LUTs | Slices | Block RAM 36K/18K(1) | DSP48E1s |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | D | 4 | 7 | 0 | 0 | 3913 | 3400 | 1708 | 55/0 | 14 |
| 1 | D | 4 | 7 | 1 | 0 | 4047 | 3351 | 1693 | 55/0 | 17 |
| 1 | D | 4 | 7 | 2 | 0 | 3906 | 3228 | 1795 | 55/0 | 14 |
| 1 | D | 4 | 7 | 1 | 1 | 5774 | 4515 | 2263 | 64/10 | 37 |
| 1 | D | 4 | 7 | 0 | 1 | 5645 | 4571 | 2265 | 64/10 | 34 |
| 1 | D | 4 | 7 | 0 | 2 | 6843 | 5964 | 2844 | 68/10 | 47 |
| 1 | D | 4 | 7 | 0 | 3 | 9189 | 7424 | 3606 | 68/10 | 78 |
| 2 | D | 4 | 7 | 0 | 3 | 10437 | 8151 | 4190 | 76/10 | 85 |
| 4 | D | 4 | 7 | 0 | 3 | 12965 | 9847 | 4961 | 92/10 | 99 |
| 8 | D | 4 | 7 | 0 | 3 | 17151 | 11670 | 6000 | 126/10 | 127 |
| 4 | D | 3 | 7 | 0 | 3 | 15640 | 11148 | 5480 | 124/10 | 115 |
| 4 | D | 2 | 7 | 0 | 3 | 15276 | 11159 | 5289 | 92/10 | 115 |
| 4 | D | 1 | 7 | 0 | 3 | 11977 | 10306 | 4315 | 124/10 | 147 |
| 2 | C | 4 | 7 | 0 | 3 | 10262 | 8341 | 3891 | 72/10 | 85 |
| 2 | B | 4 | 7 | 0 | 3 | 9946 | 7918 | 3991 | 68/10 | 85 |
| 2 | A | 4 | 7 | 0 | 3 | 9604 | 7820 | 3773 | 64/10 | 85 |
| 2 | A | 4 | 5 | 0 | 3 | 9602 | 7885 | 3601 | 64/10 | 85 |
| 2 | C | 4 | 5 | 0 | 0 | 5011 | 4024 | 2111 | 51/0 | 21 |
| 2 | C | 4 | 5 | 0 | 1 | 6754 | 5352 | 2340 | 68/10 | 41 |

**Notes:**

1. In some configurations Kintex-7 cases use a number of 18K block RAMs in addition to full 36K block RAMs.

## IP Timing Performance

The DPD IP core was characterized for resource utilization and timing performance on Virtex-6 and Kintex-7 according to the constraints shown in Table 24. These are example cases of when a single DPD IP core is placed and routed in an otherwise empty device; a user application might see different performance (better or worse). It is recommended that the user place and route the required DPD design configuration in the user application space with representative logic around it or with representative area groups for floorplanning. Contact your Xilinx Field Applications Engineer if you have timing issues or if guidance on floorplanning is required. For 8 TX cases, an area group is recommended to achieve closer to these clock frequencies.

*Table 24:* **DPD IP Timing Constraints and Timing Closure Statistics (typically for up to 4 TX)**

| | clk Speeds | proc_clk Speeds | accel_clk Speeds |
|---|---|---|---|
| Virtex-6 (cps = 1) | 333 MHz | 150 MHz | 250 |
| Kintex-7 (cps = 1) | 333 MHz | 150 MHz | 250 |
| Virtex-6 (cps = 2,3,4) | 400 MHz | 150 MHz | 250 |
| Kintex-7 (cps = 2,3,4) | 400 MHz | 150 MHz | 250 |

## SW Features Timing Performance

Figure 22 shows the per antenna execution time in seconds for the DPD ECF feature for all ARCH_SEL options and hardware acceleration (HWA) levels. The times are estimated for a MicroBlaze processor clock at 122.88MHz and an accelerator clock at 245.76MHz. LS is for execution of a COMPUTE_NEW_ COEFFICIENTS(2) control mode, or more generally ECF with DAMPEDNEWTONMU(104) equal to zero. dN is for execution of a DAMPED_UPDATE(33) control mode, or more generally ECF with DAMPEDNEWTONMU non-zero.

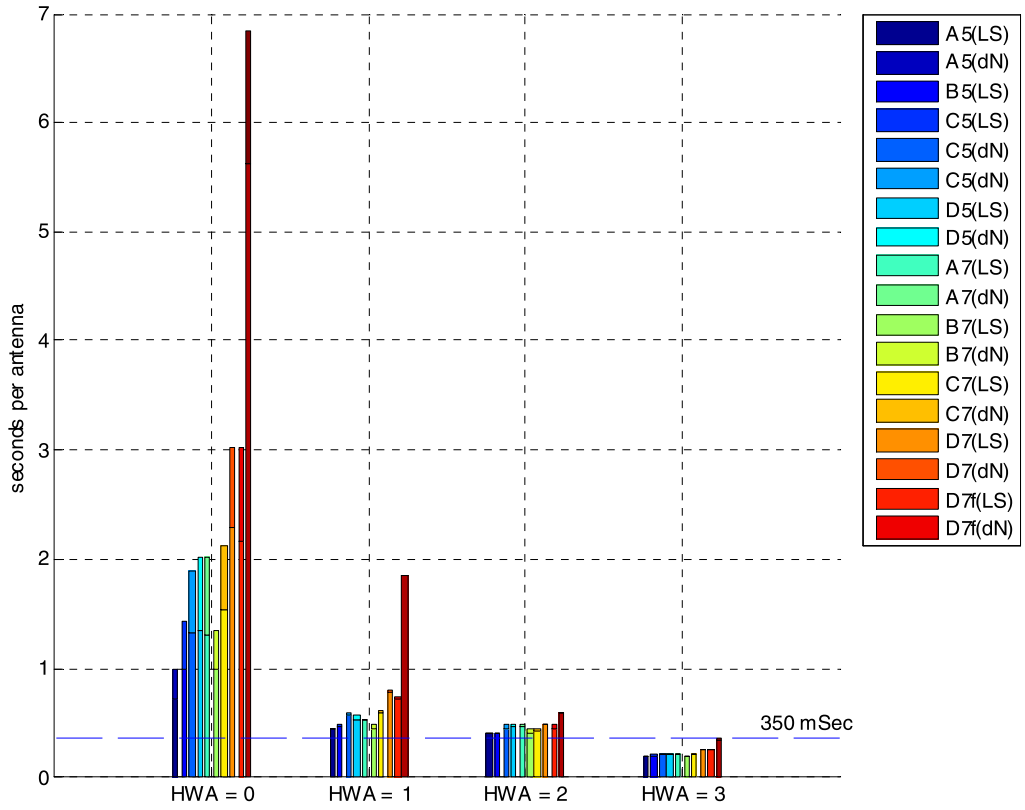


*Figure 22:* **All HWA ECF Update Time**

Figure 23 shows more detail for the hardware accelerated update times.

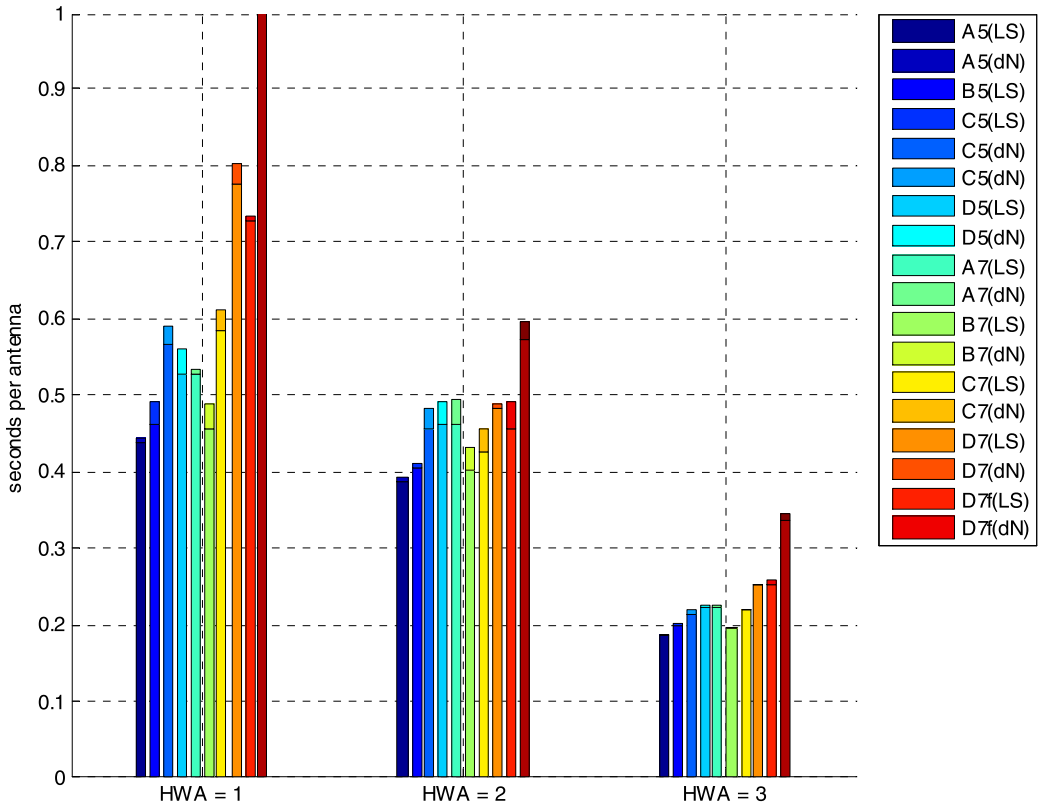


*Figure 23:* **HWA (1,2 and 3) ECF Update Time**

The HWA parameter can be used to achieve a desired overall system update rate while minimizing device utilization.

- HWA = 0
  - Software only update.
  - The update time scales linearly with clock speed.
  - Suitable for single antenna with lower complexity filters.
- HWA = 1
  - Negligible difference between dN and LS update times.
  - Architecture complexity still has an impact on update time. D7f is around 2 seconds.
- HWA = 2
  - All architectures under 0.6 second update time with less dependency on architecture complexity.
- HWA = 3
  - Reduces all updates time (including QMC) by approximately 0.25 seconds.
  - D7 update time is 250ms.
  - D7f update time is 350ms.

### Other features that impact update time

The times shown in Figure 22 are collected using the optimal $2f_s$ SRx format (RXINPUTFORMAT = 0) with RXPHASESTEP= 0.25. These times are slightly impacted when selecting different parameter configurations. The parameters that impact the update time are shown in Table 25.

*Table 25:* **Parameters Affecting Update Time**

| DPD feature | Update time impact | Notes |
|---|---|---|
| SAMPLES2PROCESS | Scales close to linearly for HWA = 0 | For non-accelerated (HWA = 0) implementations the SAMPLES2PROCESS can be reduced from the 4000 samples default, but performance should be verified. |
| Non $f_s/4$ down-conversion (RXPHASESTEP != 0.25) | +10 ms penalty | |
| | | |
| $1f_s$ SRx format (RXINPUTFORMAT = 2) | +50 ms penalty for HWA = 3 | |
| DCL cycle with QMC, per port | +250 ms penalty for HWA = 0, 1 or 2<br>+100 ms penalty for HWA = 3 | |
| FIFO ADC interface | - | Depends on implementation |

## Distortion Correction Performance

Performance testing has been conducted on the Xilinx DPD solution using an industry standard radio card, power amplifier and test equipment. The method is to instantiate the Xilinx DPD solution with supporting logic and memory that allow the transmission of data representative of any air interface standard. Test results are presented here for WCDMA, WiMAX, LTE, TD-SCDMA, CDMA2K and Multicarrier GSM. Mixed mode operation is also supported.

The hardware platform is based on the Analog Devices Mixed-Signal Digital Predistortion (MSDPD) Evaluation platform plus a Xilinx ML605 FPGA development board; spectral measurements are obtained from an Agilent E4440A PSA. The plots that follow were produced from captured trace data. For static spectra, the PA is a 3rd party LDMOS Doherty Amplifier operated with the signal peaks at P1dB (46dBm average power for 6dB PAR waveforms). Results of dynamics testing were taken with the Stealth Microwave SM2122-51LD power amplifier; the sample rate for DPD is 122.88 Msps and the observation path is a real IF sampled at 245.76 Msps.

The results that follow show the spectrum before DPD is applied and after the DCL has been run for 30 seconds. In the following results, the relative correction is shown. Whether or not a particular spectral mask requirement is met must be considered in relation to the power that the PA is being driven to. In the following results, CFR is used to maintain a PAR on the input waveform of between 6 and 7.5 dB for the FDD signals and for the maximum power segments of the TDD signals.

### WCDMA

Multicarrier data consisting of 3GPP Test Model 1 with 64 DCH is generated. Each carrier has a relative offset of 512 chips. The data is pulse-shaped, upsampled to 122.88 Msps, frequency shifted and summed.

Figure 24 and Figure 25 show the spectra before and after pre-distortion using signals that occupy a 20 MHz bandwidth, for the ARCH_SEL selections indicated, for the carrier configurations stated in the captions. Figure 26 shows 6 WCDMA carriers occupying 30MHz, this case shows an example the additional benefit of the D7f architecture with wide band signals.

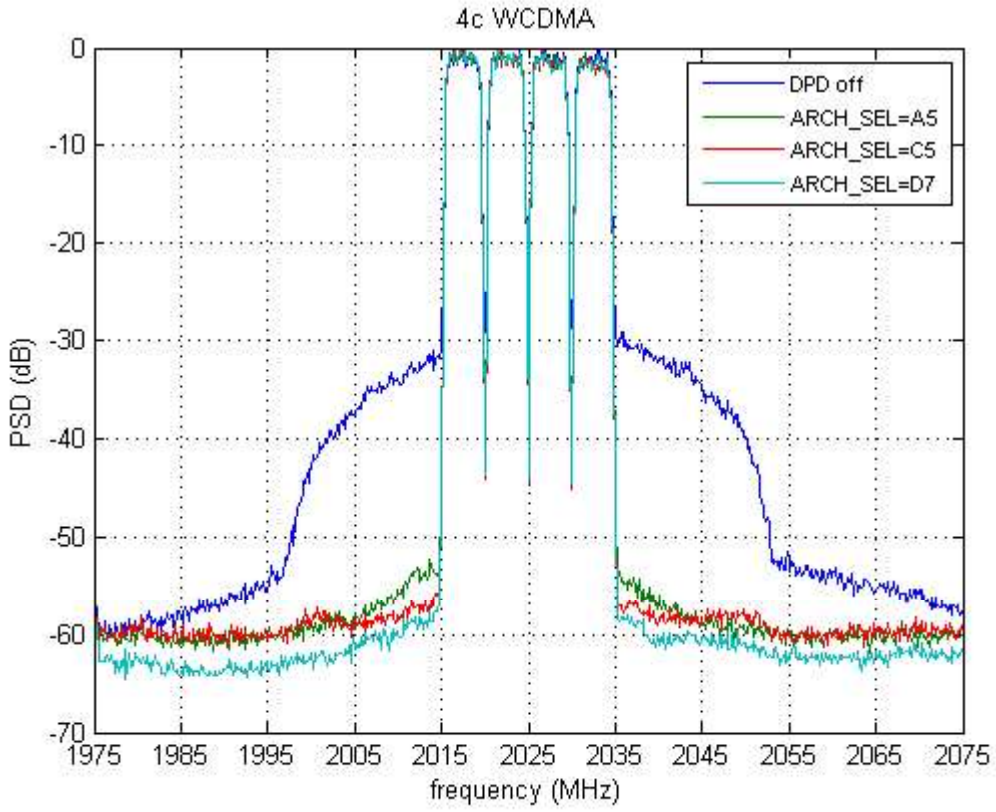


*Figure 24:* **Spectra for Four WCDMA Carriers before and after DPD**

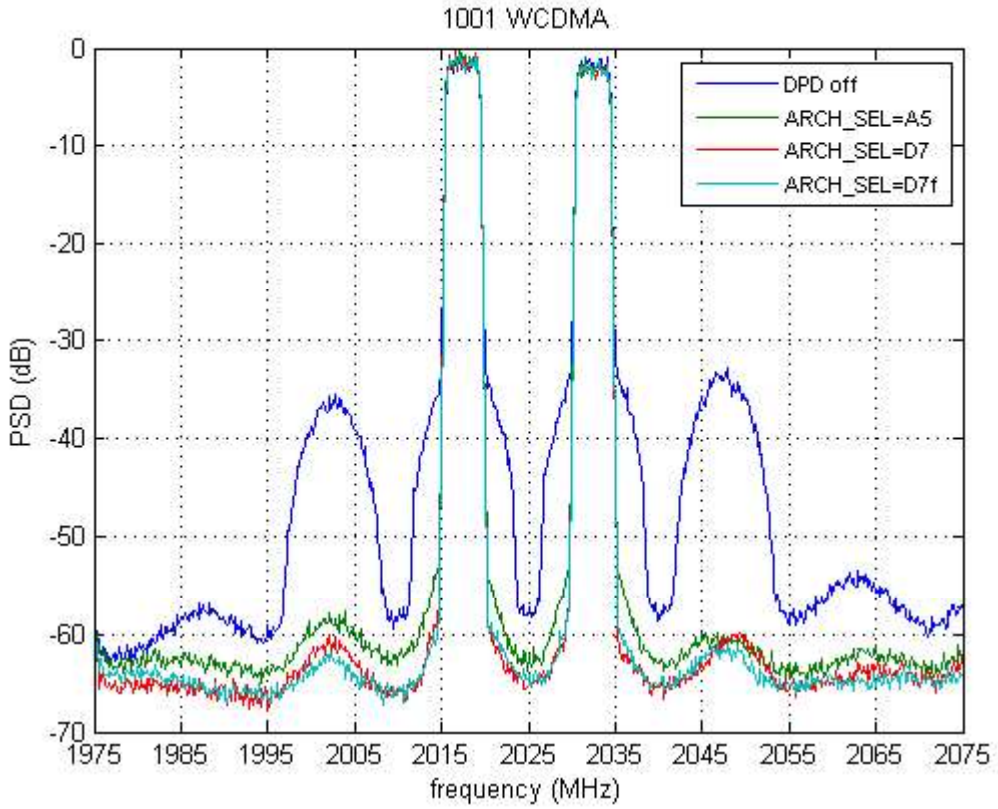


*Figure 25:* **Spectra for Two Non-adjacent WCDMA Carriers 15 MHz Apart before and after DPD**

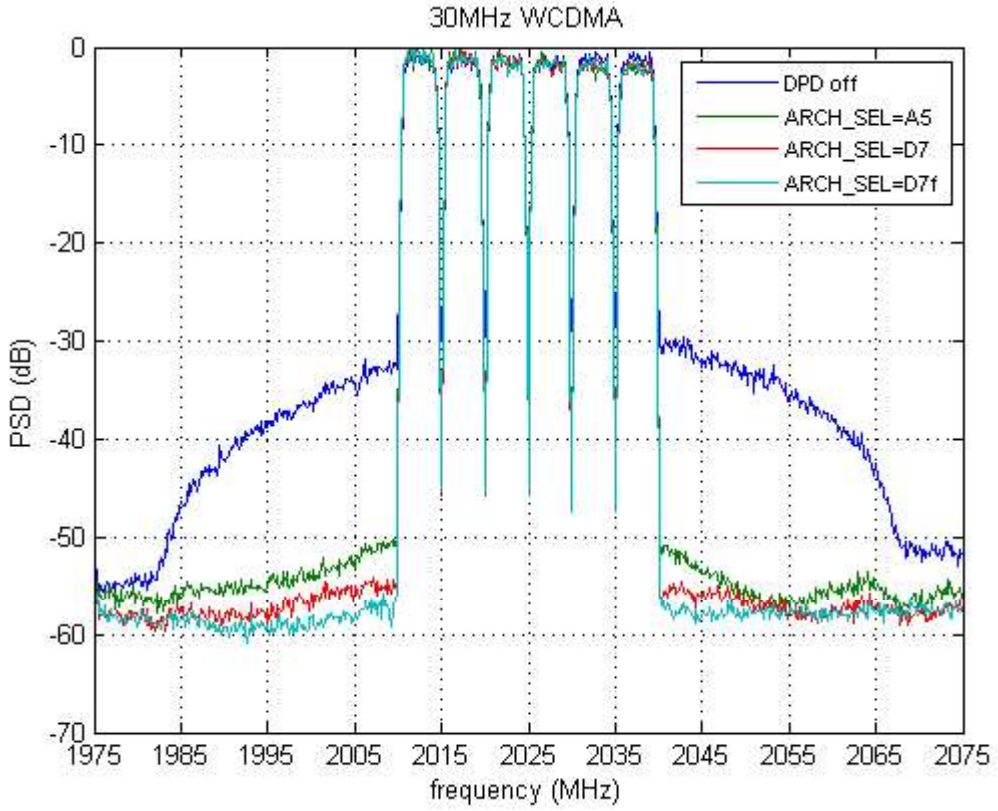


*Figure 26:* **Spectra for Six WCDMA Carriers before and after DPD**

### WiMAX

Data is generated using Agilent Signal Studio, which is standards compliant, having arbitrary zone, burst and modulation structure. It is 5 ms TDD frame data. Figure 27 shows the spectrum before and after pre-distortion for two 10 MHz carriers (one having 75% downlink active ratio and the other having 50% downlink active ratio).

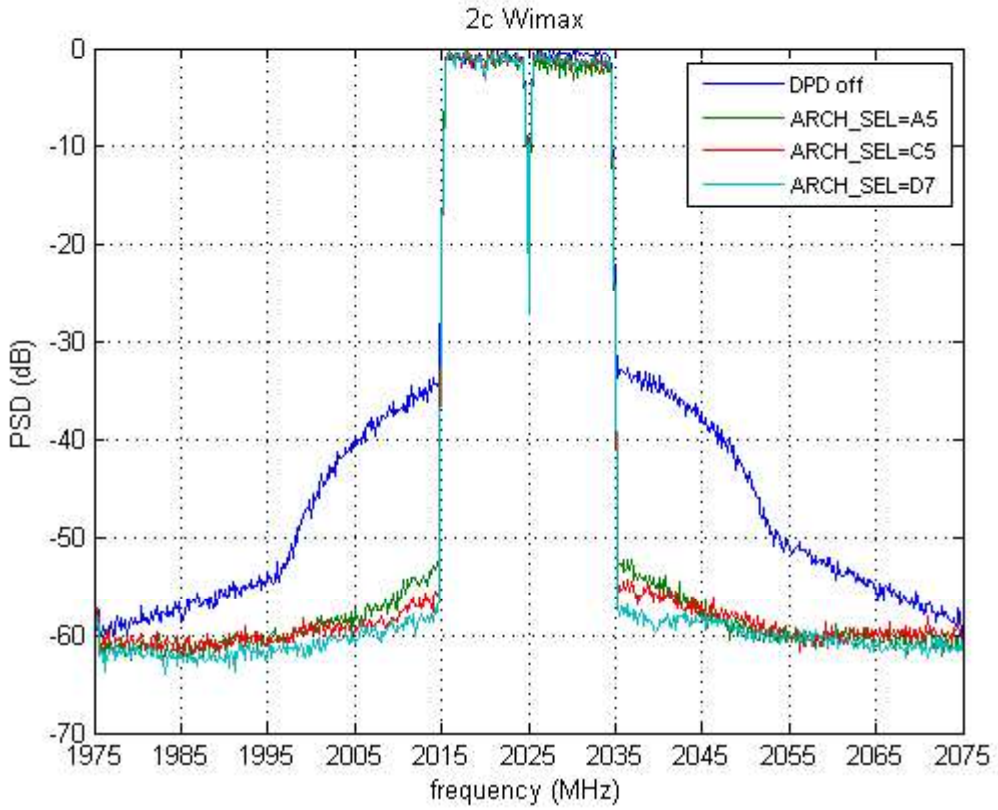


*Figure 27:* **Spectra for Two 10MHz WiMAX Carriers before and after DPD**

### LTE

Data is generated using internally developed software. The data is standards compliant with respect to frame structure and modulation. The modulation scheme is 64 QAM and the data payload is random. Figure 28 shows results for one 20 MHz carrier.

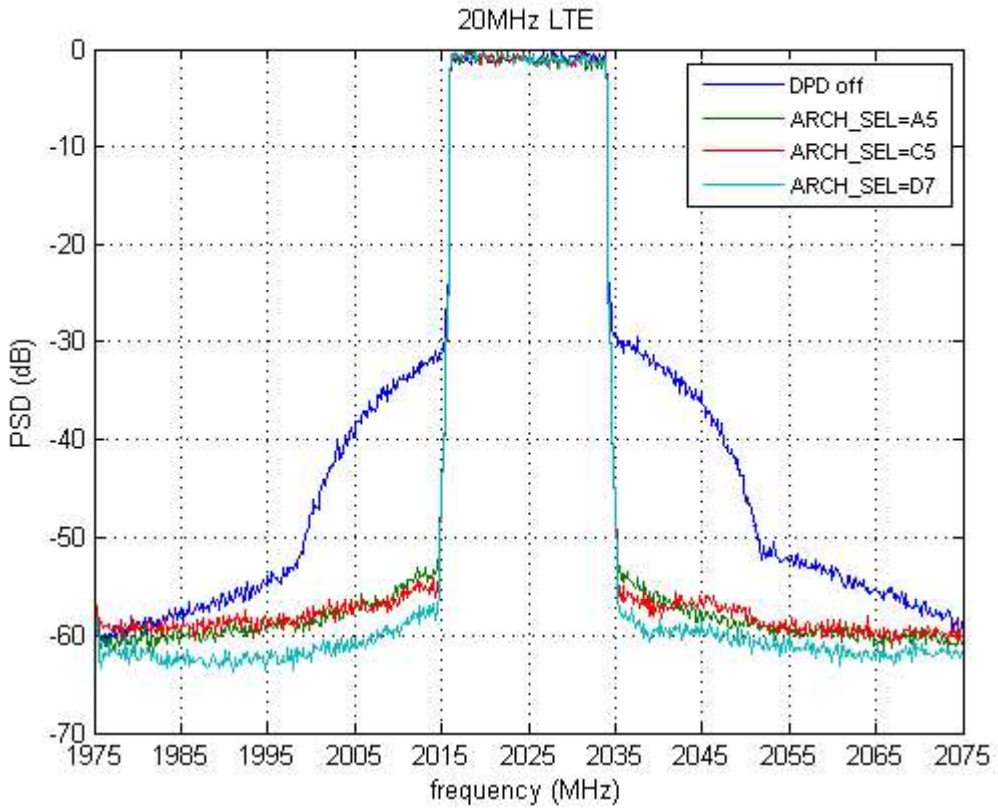


*Figure 28:* **Spectra for a Single 20MHz LTE Carrier before and after DPD**

### CDMA2000

Data is generated using internally developed software. The data is standards compliant with respect to frame structure and modulation. The data payload is random. Figure 29 shows the results for an arbitrary selection of carriers within a 20 MHz bandwidth.

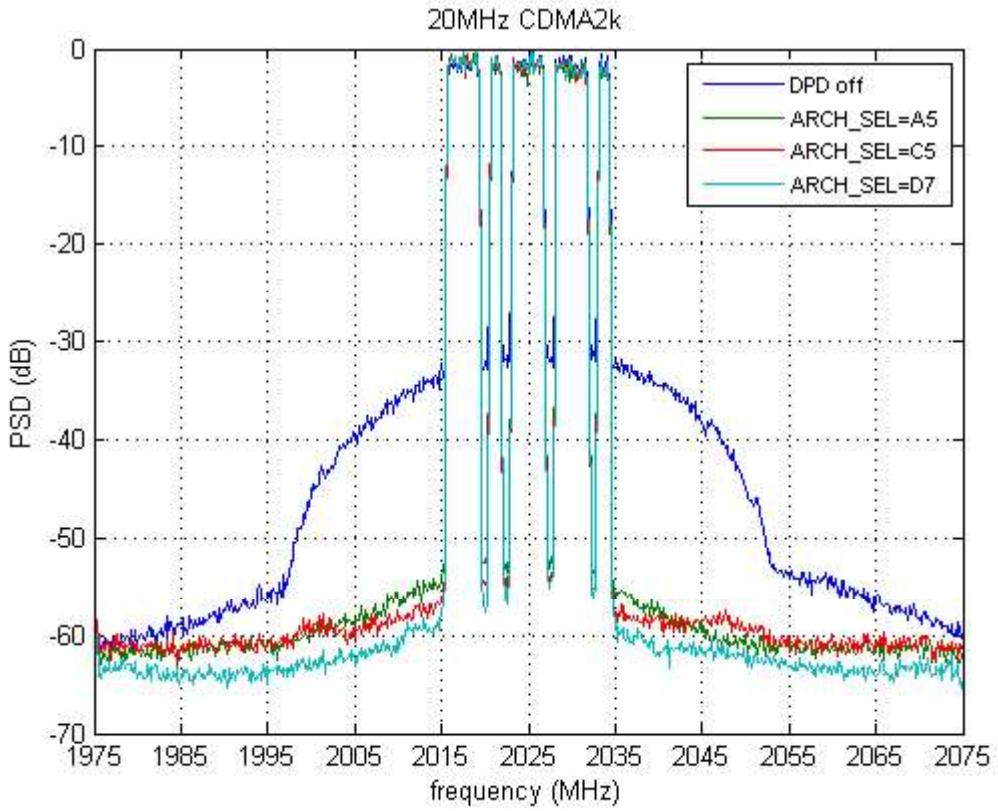


*Figure 29:* **Spectra for a 20 MHz CDMA2k Carrier before and after DPD**

### TD-SCDMA

Data is generated using internally developed software. The data is standards compliant with respect to frame structure and modulation. The data payload is random. Figure 30 shows the results for an arbitrary selection of carriers within a 15 MHz bandwidth.

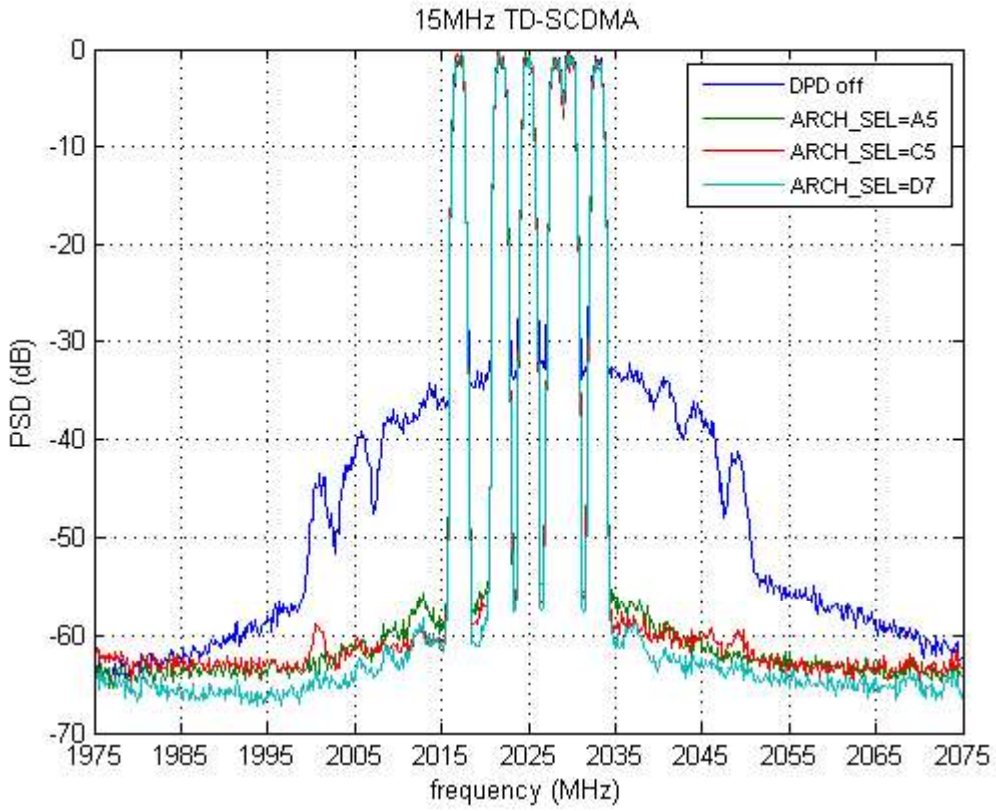


*Figure 30:* **Spectra for Six TD-SCDMA Carriers in 15MHz Total Bandwidth Carrier before and after DPD**

### Multicarrier GSM

Data is generated using internally developed software. The data is standards compliant with respect to frame structure and modulation, which is GMSK. The data payload is random. Figure 31 shows the result for an arbitrary selection of carriers within a 20 MHz bandwidth.

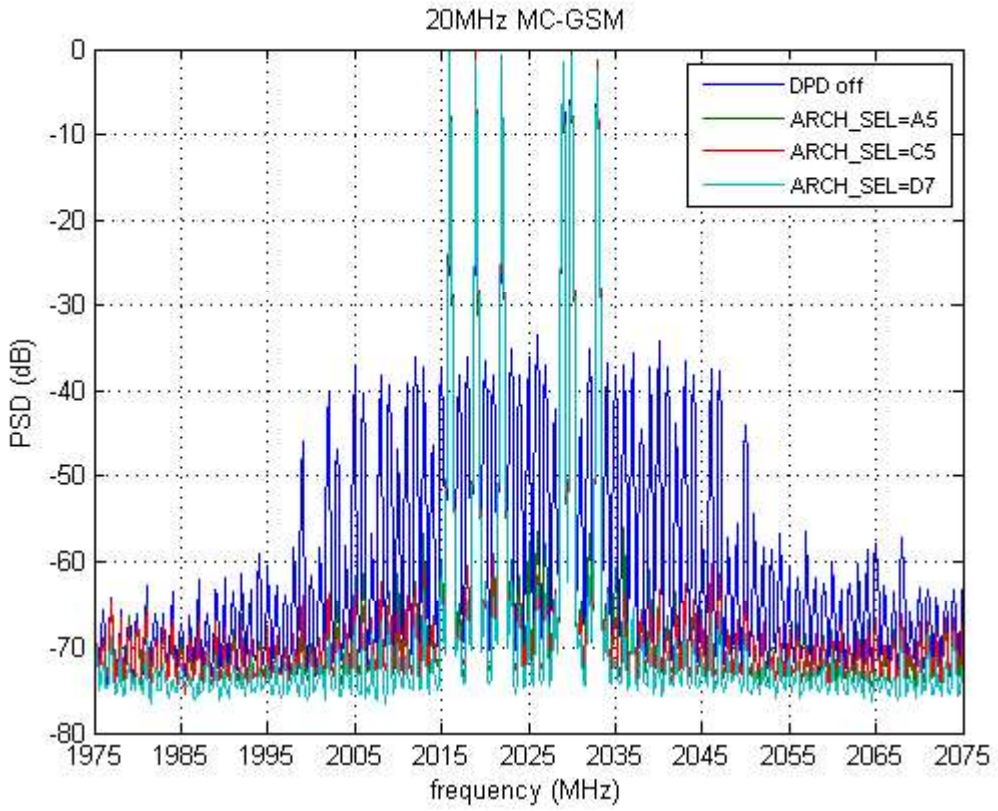


*Figure 31:* **Spectra for Four GSM Carriers in 20MHz Total Bandwidth Carrier before and after DPD**

### Dynamic and QMC Performance

Representative dynamic performance is shown in Figure 32; QMC performance is shown in Figure 33.

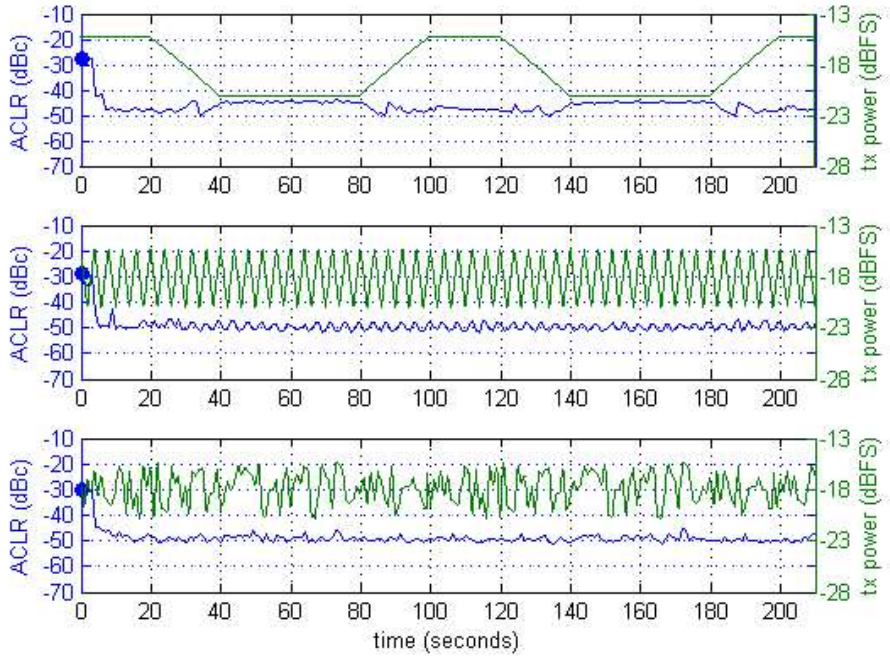


*Figure 32:* **Dynamic Performance: Adjacent Channel Ratio for Four WCDMA Carriers with Total Power Varying with Slow Steps, Fast Steps and Fast Random Profiles**

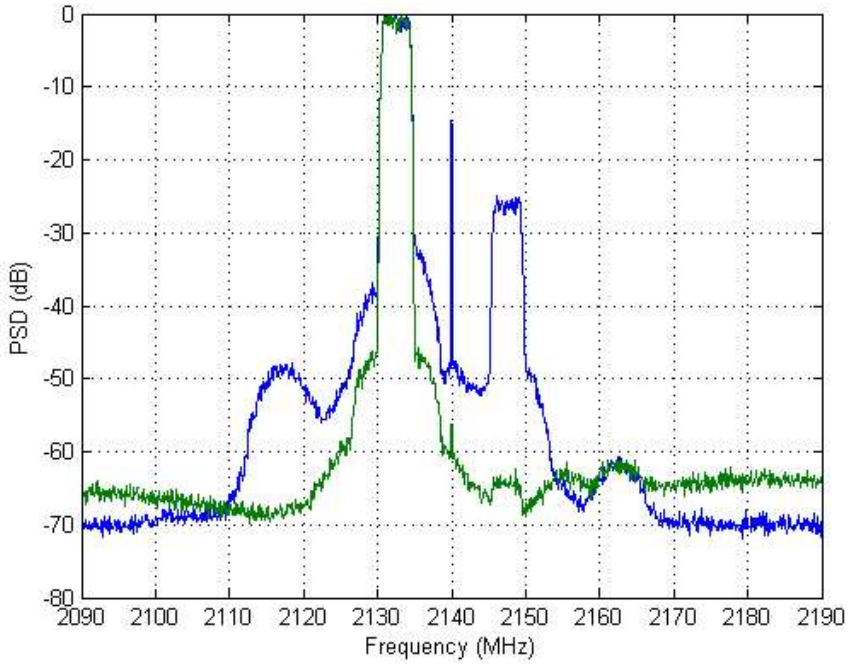


*Figure 33:* **QMC Performance: Spectra for a Single Offset WCDMA Carrier before and after QMC and DPD Correction**

## Factors Influencing Expected Correction Performance

### Overview

Because DPD is a system-level function involving subtle interactions between digital, RF and PA design, it is difficult to make hard-and-fast rules about when the demonstrated performance can be achieved. Prototyping is highly recommended, and Xilinx Support can help with the provision of evaluation platforms in certain circumstances. Nonetheless, the following sections provide some guidance.

### Sample Rates

Performance depends on the sample rate of DPD. A rule of thumb is that the pre-distortion bandwidth $f_s$ should be at least five times the signal bandwidth. However factors such as the PA design, the degree of correction required and the signal type come into play. Non-contiguous carrier configurations generally require a higher DPD sample rate than contiguous carrier configurations.

Excess pre-distortion bandwidth can also be an issue. Occasionally, wideband artifacts can be observed when $f_s$ is greater than approximately seven times the signal bandwidth, particularly if ARCH_SEL is set to 2, 3 or 4.

For optimal receive bandwidth, the DPD sample receiver rate should be exactly twice the DPD sample rate and have the signal centered in a Nyquist zone. However variances are supported. DPD can be configured for a sample receiver at one times the sample rate, and in many situations there is little performance degradation. Non-contiguous carrier configurations, however, can be particularly problematic.

There is also support for a signal not exactly centered in a Nyquist zone. If the offset is small, there might be little impact on performance.

A direct conversion receiver can also be used. In this case, QMC is unreliable unless the receiver is individually and externally calibrated.

### Required Signal Levels and Properties

The supplied design works with 16-bit transmit and up to 16-bit receive signals. The Sample Receiver (SRx) AXI4-Stream Interface section details how to use fewer bits.

In a live BTS, the transmit digital signal power varies with call load dynamics. At the maximum output power of the BTS, the scaling of the transmit signal should be such that it is optimally at -15 dBFS rms power. This value allows sufficient headroom for PAR and expansion for pre-distortion correction. Lower values can compromise the dynamic range. Some adjustment is possible, but headroom must be preserved, and scaling the signal too low can compromise performance.

DPD can, however, operate correctly at lower signal levels, but in any case the software is set to not attempt pre-distortion at signal levels less than -30 dBFS while running under DCL control (single updates are not subject to the minimum power level checks in software). Scaling to levels higher than -15 dBFS is feasible, but headroom for pre-distortion expansion and other factors should be checked explicitly.

The power in the sample receiver should be at -15 dBFS measured on the real signal, as reported by the internal DPD power meters. The signal level control is in the user's domain. The preceding test results were collected with this scaling, with a 12-bit ADC. Reasonable pre-distortion correction can be obtained with fewer ADC bits, but this should be explicitly verified in the intended installation.

CFR is helpful for good pre-distortion performance and highly recommended as a means of optimizing PA efficiency. The exact degree depends on the particular installation. The tests were conducted with approximately 6.5 dB PAR for maximum power signals (signal segments in the case of pulsed and TDD signals). The Xilinx PC-CFR v3.0 Core [Ref 1] is available for this purpose.

### RF Performance

The performance of DPD is intimately related to the quality of the RF design. The RF bandwidth should be at least five times the signal bandwidth, but special considerations might apply at the edge of the band, depending on the RF filter line-up. These are considerations outside the scope of the digital design that Xilinx offers. Within the RF bandwidth, we are unable to put limits on the amplitude and phase error that might be tolerated. Performance with RF paths worse than the Analog Devices Mixed-Signal Digital Predistortion (MSDPD) Evaluation platform is unknown.

### Parameters

The default and user-controllable settings described here normally give sufficient control for successful performance in most operational scenarios. However, DPD has a number of internal parameters and settings, and in some cases performance issues can be addressed by changing these. Xilinx Support should be contacted for assistance. See also the Support section of this document for support stipulations.

## Abbreviations

| | |
|---|---|
| 3G | Third Generation |
| 3GPP | Third Generation Partnership Project |
| ACP | Adjacent Channel Power |
| ACLR | Adjacent Carrier Leakage Ratio |
| ADC | Analog-to-Digital Converter |
| BTS | Base Transceiver Station |
| BUFG | Global Buffer (Xilinx FPGA component) |
| CAPEX | Capital Expenditure |
| CDRSX | Common Digital Radio System – Xilinx Edition |
| CFR | Crest Factor Reduction |
| CMP | Configured Maximum Power |
| CPICH | Common Pilot Channel |
| DAC | Digital-to-Analog Converter |
| dB | decibel |
| dBc | dB relative to carrier |
| dBm | dB relative to one milliwatt |
| dBFS | dB relative to digital full-scale |
| DCH | Dedicated Transport Channel |
| DCL | Dynamic Control Layer |
| DCM | Digital Clock Manager (Xilinx FPGA component) |
| DPCH | Dedicated Physical Channel |
| DPD | Digital Pre-Distortion |
| DUC | Digital Up Conversion |
| ECF | Estimation Core Function |
| FCC | Federal Communications Commission |
| FIFO | First In, First Out |
| FIR | Finite Impulse Response |

| | |
|---|---|
| HSDPA | High Speed Downlink Packet Access |
| ISR | Interrupt Service Routine |
| LDMOS | Laterally Diffused Metal Oxide Silicon (Field Effect Transistor) |
| LMB | Local Memory Bus |
| LO | Local Oscillator |
| LTE | Long Term Evolution |
| LUT | Lookup Table |
| MCP | Maximum Capacity Power |
| MSDPD | Mixed Signal Digital Pre-Distortion |
| Msps | Mega-samples per second |
| MIMO | Multiple Input Multiple Output |
| MP | Memory-Polynomial |
| NSNL | Non-Static Non-Linearity |
| ODD | Overdrive Detection |
| ODP | Overdrive Protection |
| OPEX | Operational Expenditure |
| P1dB | One-dB Compression Point |
| PA | Power Amplifier |
| PAR | Peak-to-Average Ratio |
| PLB | Peripheral Local Bus |
| QAM | Quadrature Amplitude Modulation |
| QM | Quadrature Modulator |
| QMC | Quadrature Modulator Correction |
| QSNL | Quasi-Static Non-Linearity |
| PAR | Peak-to-Average Ratio |
| RMS | Root Mean Square |
| RRC | Root-Raised Cosine |
| SA | Spectrum Analyzer |
| SBRAM | Shared Block RAM |
| SCA | Sample Capture Acceptance |
| SRx | Sample Receiver |
| TDD | Time Division Duplex |
| TD-SCDMA | Time Division Synchronous Code Division Multiple Access |
| TM1 | Test Model 1 (and similarly TM2, and so on) |
| WCDMA | Wideband Code Division Multiple Access |
| WiMAX | Worldwide Interoperability for Microwave Access |

## References

1. Xilinx Peak Cancellation Crest Factor Reduction (PC-CFR) v3.0 product page
2. LogiCORE IP Digital Pre-Distortion v4.0 Debug Interface User Guide (UG776)
3. *Xilinx AXI Design Reference Guide* (UG761)
4. AMBA 4 AXI4-Stream Protocol Version: 1.0 Specification
5. AMBA 4 AXI Protocol Specification, Version 2.0.
6. White Paper WP381: Virtex-6 FPGA Routing Optimization Design Techniques available on Xilinx website.

## Evaluation

An evaluation license is available for this core. The evaluation version operates in the same way as the full version for several hours, depending on clock frequency. The data output comprises a delayed version of the data input, after the evaluation period ends. The host interface reports EVAL_LICENSE_TIMEOUT status value (see Table 13) when the hardware times out. If you notice this behavior in hardware, it probably means you are using an evaluation version of the core. The Xilinx tools warn that an evaluation license is being used during netlist implementation. If a full license is installed, delete the old XCO file, reconfigure and regenerate the core.

## Support

Xilinx provides technical support for this LogiCORE™ product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

See the IP Release Notes Guide (XTP025) for further information on this core. There is a link to all the DSP IP and then to each core. For each core, there is a master Answer Record that contains the Release Notes and Known Issues list for each core. The following information is listed for each version of the core:

- New Features
- Bug Fixes
- Known Issues

## Ordering Information

This core can be downloaded from the Xilinx IP Center for use with the Xilinx CORE Generator software v13.2 and later. The Xilinx CORE Generator system is shipped with Xilinx ISE Design Suite development software.

To order Xilinx software, contact your local Xilinx sales representative.

Information on additional Xilinx LogiCORE IP modules is available on the Xilinx IP Center.

## Revision History

The following table shows the revision history for this document:

| Date | Version | Description of Revisions |
|---|---|---|
| 06/22/11 | 1.0 | Initial Xilinx release. ISE Design Suite 13.2. Previous version of this data sheet is DS811. |

## Notice of Disclaimer